GEEKSFORGEEKS

Process Management

1.Consider the following code fragment:

```
if (fork() == 0)
{ a = a + 5;
printf("%d,%d\n", a, &a); }
else { a = a –5; printf("%d, %d\n", a, &a); }
```

Let u, v be the values printed by the parent process, and x, y be the values printed by the child process. Which one of the following is TRUE? [C]

(A) u = x + 10 and v = y **(B)** u = x + 10 and v != y **(C)** u + 10 = x and v = y **(D)** u + 10 = x and v != y

Explanation: fork() returns 0 in child process and process ID of child process in parent process.
In Child (x), a = a + 5
In Parent (u), a = a – 5;
Therefore x = u + 10.
The physical addresses of 'a' in parent and child must be different. But our program accesses virtual addresses (assuming we are running on an OS that uses virtual memory). The child process gets an exact copy of parent process and virtual address of 'a' doesn't change in child process. Therefore, we get same addresses in both parent and child..

2.The atomic fetch-and-set x, y instruction unconditionally sets the memory location x to 1 and fetches the old value of x in y without allowing any intervening access to the memory location x. consider the following implementation of P and V functions on a binary semaphore .

```
void P (binary_semaphore *s) {
  unsigned y;
  unsigned *x = &(s->value);
  do {
    fetch-and-set x, y;
  } while (y);
}
void V (binary_semaphore *s) {
  S->value = 0;
}
```

Which one of the following is true? [A]
(A) The implementation may not work if context switching is disabled in P.
(B) Instead of using fetch-and-set, a pair of normal load/store can be used
(C) The implementation of V is wrong **(D)** The code does not implement a binary semaphore
Explanation: Let us talk about the operation P(). It stores the value of s in x, then it fetches the old value of x, stores it in y and sets x as 1. The while loop of a process will continue forever if some other process doesn't execute V() and sets the value of s as 0. If context switching is disabled in P, the while loop will run forever as no other process will be able to execute V().

3.A shared variable x, initialized to zero, is operated on by four concurrent processes W, X, Y, Z as follows. Each of the processes W and X reads x from memory, increments by one, stores it to memory, and then terminates. Each of the processes Y and Z reads x from memory, decrements by two, stores it to memory, and then terminates. Each process before reading x invokes the P operation (i.e., wait) on a counting semaphore S and invokes the V operation (i.e., signal) on the semaphore S after storing x to memory. Semaphore S is initialized to two. What is the maximum possible value of x after all processes complete execution? (GATE CS 2013) [D]

(A) -2 **(B)** -1 **(C)** 1 **(D)** 2

Explanation: Processes can run in many ways, below is one of the cases in which x attains max value
Semaphore S is initialized to 2

Process W executes S=1, x=1 but it doesn't update the x variable.

Then process Y executes S=0, it decrements x, now x= -2 and

signal semaphore S=1

Now process Z executes s=0, x=-4, signal semaphore S=1

Now process W updates x=1, S=2

Then process X executes X=2

So correct option is D

4.Three concurrent processes X, Y, and Z execute three different code segments that access and update certain shared variables. Process X executes the P operation (i.e., wait) on semaphores a, b and c; process Y executes the P operation on semaphores b, c and d; process Z executes the P operation on semaphores c, d, and a before entering the respective code segments. After completing the execution of its code segment, each process invokes the V operation (i.e., signal) on its three semaphores. All semaphores are binary semaphores initialized to one. Which one of the following represents a deadlock-free order of invoking the P operations by the processes? [B]

(A) X: P(a)P(b)P(c) Y: P(b)P(c)P(d) Z: P(c)P(d)P(a) **(B)** X: P(b)P(a)P(c) Y: P(b)P(c)P(d) Z: P(a)P(c)P(d)

(C) X: P(b)P(a)P(c) Y: P(c)P(b)P(d) Z: P(a)P(c)P(d)= **(D)** X: P(a)P(b)P(c) Y: P(c)P(b)P(d) Z: P(c)P(d)P(a)

Explanation: Option A can cause deadlock. Imagine a situation process X has acquired a, process Y has acquired b and process Z has acquired c and d. There is circular wait now.

Option C can also cause deadlock. Imagine a situation process X has acquired b, process Y has acquired c and process Z has acquired a. There is circular wait now.

Option D can also cause deadlock. Imagine a situation process X has acquired a and b, process Y has acquired c. X and Y circularly waiting for each other.

5.A shared variable x, initialized to zero, is operated on by four concurrent processes W, X, Y, Z as follows. Each of the processes W and X reads x from memory, increments by one, stores it to memory, and then terminates. Each of the processes Y and Z reads x from memory, decrements by two, stores it to memory, and then terminates. Each process before reading x invokes the P operation (i.e., wait) on a counting semaphore S and invokes the V operation (i.e., signal) on the semaphore S after storing x to memory. Semaphore S is initialized to two. What is the maximum possible value of x after all processes complete execution? [D]

(A) -2 **(B)** -1 **(C)** 1 **(D)** 2

Explanation:

Background Explanation:

A critical section in which the process may be changing common variables, updating table, writing a file and perform another function. The important problem is that if one process is executing in its critical section, no

other process is to be allowed to execute in its critical section. Each process must request permission to enter its critical section. A semaphore is a tool for synchronization and it is used to remove the critical section problem which is that no two processes can run simultaneously together so to remove this two signal operations are used named as wait and signal which is used to remove the mutual exclusion of the critical section. as an unsigned one of the most important synchronization primitives, because you can build many other Decrementing the semaphore is called acquiring or locking it, incrementing is called releasing or unlocking.

Solution :
Since initial value of semaphore is 2, two processes can enter critical section at a time- this is bad and we can see why.
Say, X and Y be the processes.X increments x by 1 and Z decrements x by 2. Now, Z stores back and after this X stores back. So, final value of x is 1 and not -1 and two Signal operations make the semaphore value 2 again. So, now W and Z can also execute like this and the value of x can be 2 which is the maximum possible in any order of execution of the processes. (If the semaphore is initialized to 1, processed would execute correctly and we get the final
value of x as -2.)

Option (D) is the correct answer.

6.A certain computation generates two arrays a and b such that a[i]=f(i) for $0 \le i < n$ and b[i]=g(a[i]) for $0 \le i < n$. Suppose this computation is decomposed into two concurrent processes X and Y such that X computes the array a and Y computes the array b. The processes employ two binary semaphores R and S, both initialized to zero. The array a is shared by the two processes. The structures of the processes are shown below.

Process X:
```
private i;
for (i=0; i < n; i++) {
  a[i] = f(i);
  ExitX(R, S);
}
```

Process Y:
```
private i;
  for (i=0; i < n; i++) {
  EntryY(R, S);
    b[i]=g(a[i]);
}
```

Which one of the following represents the CORRECT implementations of ExitX and EntryY? [C]

(A)
```
ExitX(R, S) {
  P(R);
  V(S);
}

EntryY (R, S) {
  P(S);
  V(R);
}
```

(B)
```
ExitX(R, S) {
  V(R);
  V(S);
}
```

```
EntryY(R, S) {
  P(R);
  P(S);
}
```

(C)

```
ExitX(R, S) {
  P(S);
  V(R);
}
EntryY(R, S) {
  V(S);
  P(R);
}
```

(D)

```
ExitX(R, S) {
  V(R);
  P(S);
}
EntryY(R, S) {
  V(S);
  P(R);
}
```

(A) A **(B)** B **(C)** C **(D)** D

Explanation:

The purpose here is neither the deadlock should occur nor the binary semaphores be assigned value greater than one.

A leads to deadlock

B can increase value of semaphores b/w 1 to n

D may increase the value of semaphore R and S to 2 in some cases

7.Fetch_And_Add(X,i) is an atomic Read-Modify-Write instruction that reads the value of memory location X, increments it by the value i, and returns the old value of X. It is used in the pseudocode shown below to

implement a busy-wait lock. L is an unsigned integer shared variable initialized to 0. The value of 0 corresponds to lock being available, while any non-zero value corresponds to the lock being not available.

```
AcquireLock(L){
    while (Fetch_And_Add(L,1))
        L = 1;
}
ReleaseLock(L){
    L = 0;
}
```

This implementation [B]
(A) fails as L can overflow **(B)** fails as L can take on a non-zero value when the lock is actually available
(C) works correctly but may starve some processes **(D)** works correctly without starvation
Explanation: Take closer look the below while loop.

```
while (Fetch_And_Add(L,1))
    L = 1;  // A waiting process
        // can be here just after
        // the lock is released,
        // and can make L = 1.
```

Consider a situation where a process has just released the lock and made L = 0. Let there be one more process waiting for the lock, means executing the AcquireLock() function. Just after the L was made 0, let the waiting processes executed the line L = 1. Now, the lock is available and L = 1. Since L is 1, the waiting process (and any other future coming processes) can not come out of the while loop.

The above problem can be resolved by changing the AcuireLock() to following.

```
AcquireLock(L){
    while (Fetch_And_Add(L,1))
    { // Do Nothing }
}
```

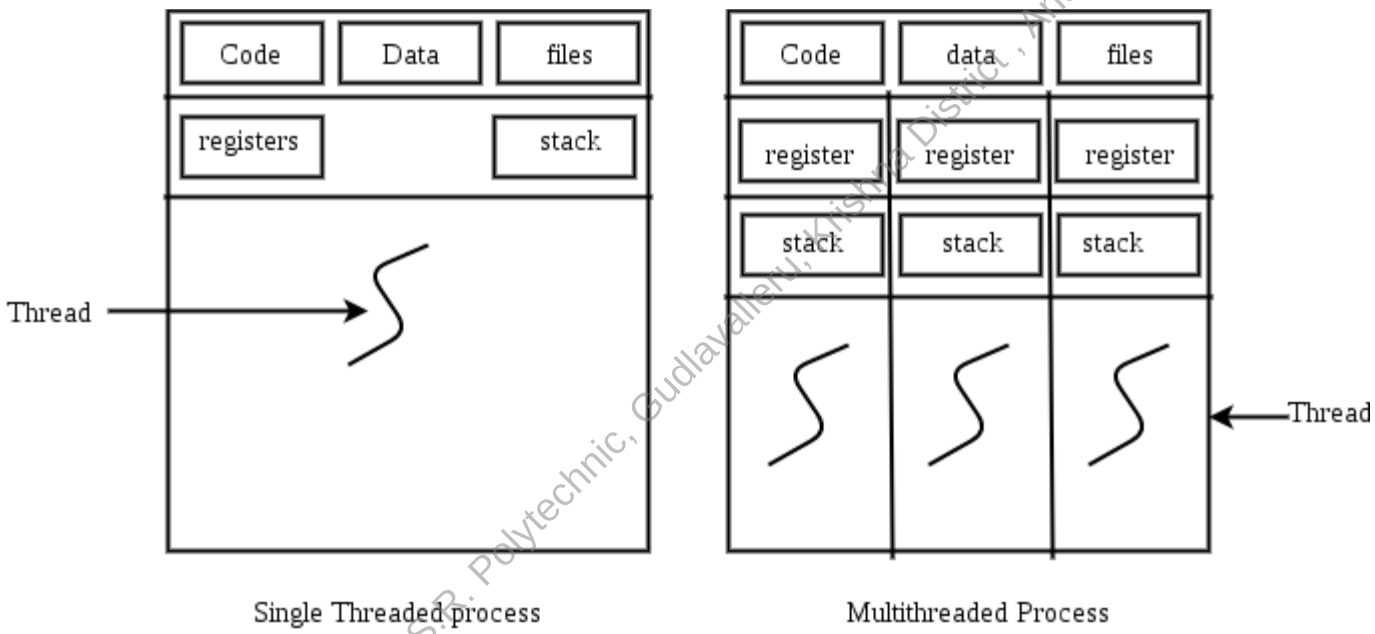
Overflow can occur only when larger number of processes than size of 'int' execute the check condition of while loop but not L = 1, i.e., preemption required.
But when L is 1, the process repeats in the while loop- there is no overflow because after each increment to L, L is again made equal to 1.
Option (b) is best choice over option (a).

9.The time taken to switch between user and kernel modes of execution be t1 while the time taken to switch between two processes be t2.

Which of the following is TRUE? [C]
(A) t1 > t2 **(B)** t1 = t2 **(C)** t1 < t2
(D) nothing can be said about the relation between t1 and t2
Explanation: Process switches or Context switches can occur in only kernel mode . So for process switches first we have to move from user to kernel mode . Then we have to save the PCB of the process from which we are taking off CPU and then we have to load PCB of the required process . At switching from kernel to user

mode is done. But switching from user to kernel mode is a very fast operation(OS has to just change single bit at hardware level)
Thus T1< T2

10.A thread is usually defined as a "light weight process" because an operating system (OS) maintains smaller data structures for a thread than for a process. In relation to this, which of the following is TRUE? [C]
(A) On per-thread basis, the OS maintains only CPU register state
(B) The OS does not maintain a separate stack for each thread
(C) On per-thread basis, the OS does not maintain virtual memory state
(D) On per-thread basis, the OS maintains only scheduling and accounting information
Explanation: Threads share address space of Process. Virtually memory is concerned with processes not with Threads.
A thread is a basic unit of CPU utilization, consisting of a program counter, a stack, and a set of registers, (and a thread ID.) As you can see, for a single thread of control – there is one program counter, and one sequence of instructions that can be carried out at any given time and for multi-threaded applications-there are multiple threads within a single process, each having their own program counter, stack and set of registers, but sharing common code, data, and certain structures such as open files.



Option (A): as you can see in the above diagram, NOT ONLY CPU Register but stack and code files, data files are also maintained. So, option (A) is not correct as it says OS maintains only CPU register state.
Option (B): according to option (B), OS does not maintain a separate stack for each thread. But as you can see in above diagram, for each thread, separate stack is maintained. So this option is also incorrect.

Option (C): according to option (C), the OS does not maintain virtual memory state. And It is correct as Os does not maintain any virtual memory state for individual thread.
Option (D): according to option (D), the OS maintains only scheduling and accounting information. But it is not correct as it contains other information like cpu registers stack, program counters, data files, code files are also maintained.
12.Consider the methods used by processes P1 and P2 for accessing their critical sections whenever needed, as given below. The initial values of shared boolean variables S1 and S2 are randomly assigned.

Method Used by P1

```
while (S1 == S2) ;
```

Critica1 Section

S1 = S2;

Method Used by P2

while (S1 != S2) ;

Critica1 Section

S2 = not (S1);

Which one of the following statements describes the properties achieved? [A]

(A) Mutual exclusion but not progress **(B)** Progress but not mutual exclusion

(C) Neither mutual exclusion nor progress **(D)** Both mutual exclusion and progress

Explanation:

Mutual Exclusion:

A way of making sure that if one process is using a shared modifiable data, the other processes will be excluded from doing the same thing. while one process executes the shared variable, all other processes desiring to do so at the same time moment should be kept waiting; when that process has finished executing the shared variable, one of the processes waiting; while that process has finished executing the shared variable, one of the processes waiting to do so should be allowed to proceed. In this fashion, each process executing the shared data (variables) excludes all others from doing so simultaneously. This is called Mutual Exclusion.

Progress Requirement:

If no process is executing in its critical section and there exist some processes that wish to enter their critical section, then the selection of the processes that will enter the critical section next cannot be postponed indefinitely.

Solution:

It can be easily observed that the Mutual Exclusion requirement is satisfied by the above solution, P1 can enter critical section only if S1 is not equal to S2, and P2 can enter critical section only if S1 is equal to S2. But here Progress Requirement is not satisfied. Suppose when s1=1 and s2=0 and process p1 is not interested to enter into critical section but p2 want to enter critical section. P2 is not able to enter critical section in this as only when p1 finishes execution, then only p2 can enter (then only s1 = s2 condition be satisfied). Progress will not be satisfied when any process which is not interested to enter into the critical section will not allow other interested process to enter into the critical section.

13.The following program consists of 3 concurrent processes and 3 binary semaphores.The semaphores are initialized as S0 = 1, S1 = 0, S2 = 0.

Process P0	Process P1	Process P2

while (true) {
wait (S0);
print '0';
release (S1);
release (S2);
}
	wait (S1);
	release (S0);
		wait (S2)
		release (S0);

How many times will process P0 print '0'? [A]

(A) At least twice **(B)** Exactly twice **(C)** Exactly thrice **(D)** Exactly once

Explanation: Initially only P0 can go inside the while loop as S0 = 1, S1 = 0, S2 = 0.

Minimum no. of time 0 printed is twice when execute in this order (p0 -> p1 -> p2 -> p0).Maximum no. of time 0 printed is thrice when execute in this order (p0 -> p1 -> p0 -> p2 -> p0).The enter_CS() and leave_CS() functions to implement critical section of a process are realized using test-and-set instruction as follows:

```
void enter_CS(X)
{
   while test-and-set(X) ;
}
void leave_CS(X)
{
  X = 0;
}
```

14.In the above solution, X is a memory location associated with the CS and is initialized to 0. Now consider the following statements:
I. The above solution to CS problem is deadlock-free II. The solution is starvation free.
III. The processes enter CS in FIFO order. IV More than one process can enter CS at the same time.

Which of the above statements is TRUE? [A]

(A) I only **(B)** I and II **(C)** II and III **(D)** IV only

Explanation: The above solution is a simple test-and-set solution that makes sure that deadlock doesn't occur, but it doesn't use any queue to avoid starvation or to have FIFO order.

15.The P and V operations on counting semaphores, where s is a counting semaphore, are defined as follows:

```
P(s) : s =  s - 1;
if (s  < 0) then wait;
V(s) : s = s + 1;
if (s <= 0) then wakeup a process waiting on s;
```

Assume that Pb and Vb the wait and signal operations on binary semaphores are provided. Two binary semaphores Xb and Yb are used to implement the semaphore operations P(s) and V(s) as follows:

```
P(s) : Pb(Xb);
s = s - 1;
if (s < 0) {
Vb(Xb) ;
Pb(Yb) ;
}
else Vb(Xb);
V(s) : Pb(Xb) ;
s = s + 1;
if (s <= 0) Vb(Yb) ;
Vb(Xb) ;
```

The initial values of Xb and Yb are respectively [C]

(A) 0 and 0 **(B)** 0 and 1 **(C)** 1 and 0 **(D)** 1 and 1

Explanation: Suppose Xb = 0, then because of P(s): Pb(Xb) operation, Xb will be -1 and processs will get blocked as it will enter into waiting section.
So, Xb will be one.

Suppose s=2(means 2 process are accessing shared resource), taking Xb as 1,

first P(s): Pb(Xb) operation will make Xb as zero. s will be 1 and Then Vb(Xb) operation will be executed which will increase the count of Xb as one. Then same process will be repeated making Xb as one and s as zero. Now suppose one more process comes, then Xb will be 0 but s will be -1 which will make this process go into loop (s <0) and will result into calling Vb(Xb) and Pb(Yb) operations. Vb(Xb) will result into Xb as 1 and Pb(Yb) will result into decrementing the value of Yb.

case 1: if Yb has value as 0, it will be -1 and it will go into waiting and will be blocked.total 2 process will access shared resource (according to counting semaphore, max 3 process can access shared resource) and value of s is -1 means only 1 process will be waiting for resources and just now, one process got blocked. So it is still true.
case 2: if Yb has value as 1, it will be 0. Total 3 process will access shared resource (according to counting semaphore, max 2 process can access shared resource) and value of s is -1 means only 1 process will be waiting for resources and but there is no process waiting for resources.So it is false.
16.A process executes the following code

```
for (i = 0; i < n; i++) fork();
```

The total number of child processes created is [B]
(A) n **(B)** $2^n - 1$ **(C)** 2^n **(D)** $2^{(n+1)} - 1$
17.If we sum all levels of above tree for i = 0 to n-1, we get $2^n - 1$. So there will be $2^n - 1$ child processes. On the other hand, the total number of **process** created are (number of child processes)+1.
Consider the following statements about user level threads and kernel level threads. Which one of the following statement is FALSE? [D]
(A) Context switch time is longer for kernel level threads than for user level threads.
(B) User level threads do not need any hardware support.
(C) Related kernel level threads can be scheduled on different processors in a multi-processor system.
(D) Blocking one kernel level thread blocks all related threads.
Explanation:
Kernel level threads are managed by the OS, therefore, thread operations are implemented in the kernel code. Kernel level threads can also utilize multiprocessor systems by splitting threads on different processors. If one thread blocks it does not cause the entire process to block. Kernel level threads have disadvantages as well. They are slower than user level threads due to the management overhead. Kernel level context switch involves more steps than just saving some registers. Finally, they are not portable because the implementation is operating system dependent.
option (A): Context switch time is longer for kernel level threads than for user level threads. True, As User level threads are managed by user and Kernel level threads are managed by OS. There are many overheads involved in Kernel level thread management, which are not present in User level thread management. So context switch time is longer for kernel level threads than for user level threads.

Option (B): User level threads do not need any hardware support True, as User level threads are managed by user and implemented by Libraries, User level threads do not need any hardware support.

Option (C): Related kernel level threads can be scheduled on different processors in a multi- processor system. This is true.

Option (D): Blocking one kernel level thread blocks all related threads. false, since kernel level threads are managed by operating system, if one thread blocks, it does not cause all threads or entire process to block.

18.Two processes, P1 and P2, need to access a critical section of code. Consider the following synchronization construct used by the processes:Here, wants1 and wants2 are shared variables, which are initialized to false. Which one of the following statements is TRUE about the above construct? [D]

```
while (true) {
```

```
wants1 = true;
while (wants2 == true);
/* Critical Section */
wants1=false;
}
/* Remainder section */
/* P2 */
while (true) {
wants2 = true;
while (wants1==true);
/* Critical Section */
wants2 = false;
}
/* Remainder section */
```

(A) It does not ensure mutual exclusion. **(B)** It does not ensure bounded waiting.
(C) It requires that processes enter the critical section in strict alternation.
(D) It does not prevent deadlocks, but ensures mutual exclusion.

Explanation:

Bounded waiting :There exists a bound, or limit, on the number of times other processes are allowed to enter their critical sections after a process has made request to enter its critical section and before that request is granted.

mutual exclusion prevents simultaneous access to a shared resource. This concept is used in concurrent programming with a critical section, a piece of code in which processes or threads access a shared resource.

Solution:

Two processes, P1 and P2, need to access a critical section of code. Here, wants1 and wants2 are shared variables, which are initialized to false.

Now, when both wants1 and wants2 become true, both process p1 and p2 enter in while loop and waiting for each other to finish. This while loop run indefinitely which leads to deadlock.

19.Now, Assume P1 is in critical section (it means wants1=true, wants2 can be anything, true or false). So this ensures that p2 won't enter in critical section and vice versa. This satisfies the property of mutual exclusion. Here bounded waiting condition is also satisfied as there is a bound on the number of process which gets access to critical section after a process request access to it.

Which one of the following is FALSE? [D]

(A) User level threads are not scheduled by the kernel.
(B) When a user level thread is blocked, all other threads of its process are blocked.
(C) Context switching between user level threads is faster than context switching between kernel level threads. **(D)** Kernel level threads cannot share the code segment

Explanation:

User level thread	**Kernel level thread**
User thread are implemented by user processes.	kernel threads are implemented by OS.

OS doesn't recognized user level threads.	Kernel threads are recognized by OS.
Implementation of User threads is easy.	Implementation of Kernel thread is complicated.
Context switch time is less.	**Context switch time is more.**
Context switch requires no hardware support.	Hardware support is needed.
If one user level thread perform blocking operation then entire process will be blocked.	If one kernel thread perform blocking operation then another thread can continue execution.
Example : Java thread, POSIX threads.	Example : Window Solaris.

20.Consider two processors P1 and P2 executing the same instruction set. Assume that under identical conditions, for the same input, a program running on P2 takes 25% less time but incurs 20% more CPI (clock cycles per instruction) as compared to the program running on P1. If the clock frequency of P1 is 1GHz, then the clock frequency of P2 (in GHz) is _____. [A]

(A) 1.6 **(B)** 3.2 **(C)** 1.2 **(D)** 0.8

Explanation:

For P1 clock period = 1ns

Let clock period for P2 be t.

Now consider following equation based on specification

7.5 ns = 12*t ns

We get t and inverse of t will be 1.6GHz

Consider the procedure below for the Producer-Consumer problem which uses semaphores:

```
semaphore n = 0;
semaphore s = 1;
void producer()
{
      while(true)
      {
       produce();
       semWait(s);
       addToBuffer();
       semSignal(s);
       semSignal(n);
      }
}
```

```
void consumer()
{
      while(true)
      {
       semWait(s);
       semWait(n);
       removeFromBuffer();
       semSignal(s);
       consume();
      }
}
```

21.Which one of the following is TRUE? [A]

(A) The producer will be able to add an item to the buffer, but the consumer can never consume it.

(B) The consumer will remove no more than one item from the buffer.

(C) Deadlock occurs if the consumer succeeds in acquiring semaphore s when the buffer is empty.

(D) The starting value for the semaphore n must be 1 and not 0 for deadlock-free operation.

Explanation: Initially, there is no element in the buffer.

Semaphore *s=1* and semaphore *n=0*.

We assume that initially control goes to the consumer when buffer is empty.

semWait(s) decrements the value of semaphore 's' . Now, s = 0 and *semWait(n)* decrements the value of semaphore 'n'.
Since, the value of semaphore 'n' becomes less than 0 , the control stucks in while loop of function semWait() and a deadlock arises.
Thus, deadlock occurs if the consumer succeeds in acquiring semaphore s when the buffer is empty.
22.The atomic fetch-and-set x, y instruction unconditionally sets the memory location x to 1 and fetches the old value of x n y without allowing any intervening access to the memory location x. consider the following implementation of P and V functions on a binary semaphore S.

```
void P (binary_semaphore *s)
{
   unsigned y;
   unsigned *x = &(s->value);
   do
   {
      fetch-and-set x, y;
   }
   while (y);
}
void V (binary_semaphore *s)
{
   S->value = 0;
}
```

Which one of the following is true? [A]
(A) The implementation may not work if context switching is disabled in P
(B) Instead of using fetch-and –set, a pair of normal load/store can be used
(C) The implementation of V is wrong **(D)** The code does not implement a binary semaphore
Barrier is a synchronization construct where a set of processes synchronizes globally i.e. each process in the set arrives at the barrier and waits for all others to arrive and then all processes leave the barrier. Let the number of processes in the set be three and S be a binary semaphore with the usual P and V functions. Consider the following C implementation of a barrier with line numbers shown on left.

```
void barrier (void) {
1:  P(S);
2:  process_arrived++;
3.  V(S);
4:  while (process_arrived !=3);
```

```
5:  P(S);
6:  process_left++;
7:  if (process_left==3) {
8:     process_arrived = 0;
9:     process_left = 0;
10: }
11: V(S);
}
```

23.The variables process_arrived and process_left are shared among all processes and are initialized to zero. In a concurrent program all the three processes call the barrier function when they need to synchronize globally.

The above implementation of barrier is incorrect. Which one of the following is true? [B]
(A) The barrier implementation is wrong due to the use of binary semaphore S
(B) The barrier implementation may lead to a deadlock if two barrier in invocations are used in immediate succession. **(C)** Lines 6 to 10 need not be inside a critical section
(D) The barrier implementation is correct if there are only two processes instead of three.
Explanation: It is possible that process_arrived becomes greater than 3. It will not be possible for process arrived to become 3 again, hence deadlock.
Barrier is a synchronization construct where a set of processes synchronizes globally i.e. each process in the set arrives at the barrier and waits for all others to arrive and then all processes leave the barrier. Let the number of processes in the set be three and S be a binary semaphore with the usual P and V functions. Consider the following C implementation of a barrier with line numbers shown on left.


```
void barrier (void) {
1:  P(S);
2:  process_arrived++;
3.  V(S);
4:  while (process_arrived !=3);
5:  P(S);
6:  process_left++;
```

7: if (process_left==3) {

8: process_arrived = 0;

9: process_left = 0;

10: }

11: V(S);

}

24.The variables process_arrived and process_left are shared among all processes and are initialized to zero. In a concurrent program all the three processes call the barrier function when they need to synchronize globally.Which one of the following rectifies the problem in the implementation? [B]

(A) Lines 6 to 10 are simply replaced by process_arrived–

(B) At the beginning of the barrier the first process to enter the barrier waits until process_arrived becomes zero before proceeding to execute P(S).

(C) Context switch is disabled at the beginning of the barrier and re-enabled at the end.

(D) The variable process_left is made private instead of shared.

Explanation:

Step '2' should not be executed when the process enters the barrier second time till other two processes have not completed their 7th step. This is to prevent variable process_arrived becoming greater than 3.

So, when variable process_arrived becomes zero and variable process_left also becomes zero then the problem of deadlock will be resolved.Thus, at the beginning of the barrier the first process to enter the barrier waits until process_arrived becomes zero before proceeding to execute P(S).

25.Consider two processes P1 and P2 accessing the shared variables X and Y protected by two binary semaphores SX and SY respectively, both initialized to 1. P and V denote the usual semaphone operators, where P decrements the semaphore value, and V increments the semaphore value. The pseudo-code of P1 and P2 is as follows :

P1 :

While true do {

L1 : .....

L2 : .....

X = X + 1;

Y = Y - 1;

V(SX);

V(SY);}

P2 :

While true do {

L3 : .....

L4 : .....

Y = Y + 1;

X = Y - 1;

V(SY);

V(SX); }

In order to avoid deadlock, the correct operators at L1, L2, L3 and L4 are respectively [D]

(A) P(SY), P(SX); P(SX), P(SY) **(B)** P(SX), P(SY); P(SY), P(SX)

(C) P(SX), P(SX); P(SY), P(SY) **(D)** P(SX), P(SY); P(SX), P(SY)

Suppose we want to synchronize two concurrent processes P and Q using binary semaphores S and T. The code for the processes P and Q is shown below.

Process P:

```
while (1) {
W:
  print '0';
  print '0';
X:
}
```

Process Q:

```
while (1) {
Y:
  print '1';
  print '1';
Z:
}
```

25.Synchronization statements can be inserted only at points W, X, Y and Z. ***Which of the following will always lead to an output staring with '001100110011' ?*** ***[B]***

(A) P(S) at W, V(S) at X, P(T) at Y, V(T) at Z, S and T initially 1

(B) P(S) at W, V(T) at X, P(T) at Y, V(S) at Z, S initially 1, and T initially 0

(C) P(S) at W, V(T) at X, P(T) at Y, V(S) at Z, S and T initially 1

(D) P(S) at W, V(S) at X, P(T) at Y, V(T) at Z, S initially 1, and T initially 0

Explanation: P(S) means wait on semaphore 'S' and V(S) means signal on semaphore 'S'.

Wait(S)

{ while (i <= 0)

--S;

}

Signal(S) {

S++;}

Initially, we assume S = 1 and T = 0 to support mutual exclusion in process P and Q.
Since S = 1, only process P will be executed and wait(S) will decrement the value of S. Therefore, S = 0.
At the same instant, in process Q, value of T = 0. Therefore, in process Q, control will be stuck in while loop till the time process P prints 00 and increments the value of T by calling the function V(T).
While the control is in process Q, semaphore S = 0 and process P would be stuck in while loop and would not execute till the time process Q prints 11 and makes the value of S = 1 by calling the function V(S).
This whole process will repeat to give the output 00 11 00 11 … .
Suppose we want to synchronize two concurrent processes P and Q using binary semaphores S and T. The code for the processes P and Q is shown below.

Process P:

```
while (1) {
W:
  print '0';
  print '0';
X:
}
```

Process Q:

```
while (1) {
Y:
  print '1';
  print '1';
Z:
}
```

26.Synchronization statements can be inserted only at points W, X, Y and Z

Which of the following will ensure that the output string never contains a substring of the form 01^n0 or 10^n1 where n is odd? [C]
(A) P(S) at W, V(S) at X, P(T) at Y, V(T) at Z, S and T initially 1
(B) P(S) at W, V(T) at X, P(T) at Y, V(S) at Z, S and T initially 1
(C) P(S) at W, V(S) at X, P(S) at Y, V(S) at Z, S initially 1
(D) V(S) at W, V(T) at X, P(S) at Y, P(T) at Z, S and T initially 1
Explanation: P(S) means wait on semaphore 'S' and V(S) means signal on semaphore 'S'. The definition of these functions are :

```
 Wait(S)  {
while (i <= 0) ;
S-- ; }
Signal(S)  {
S++ ;
}
```

Initially S = 1 and T = 0 to support mutual exclusion in process 'P' and 'Q'.
Since, S = 1 , process 'P' will be executed and function Wait(S) will decrement the value of 'S'. So, S = 0 now.Simultaneously, in process 'Q' , T = 0 . Therefore, in process 'Q' control will be stuck in while loop till the time process 'P' prints '00' and increments the value of 'T' by calling function V(T).While the control is in process 'Q', S = 0 and process 'P' will be stuck in while loop. Process 'P' will not execute till the time process 'Q' prints '11' and makes S = 1 by calling function V(S).Thus, process 'P' and 'Q' will keep on repeating to give the output '00110011 …… ' .

27.Which of the following does not interrupt a running process? [C]
(A) A device **(B)** Timer **(C)** Scheduler process **(D)** Power failure
Explanation: Scheduler process doesn't interrupt any process, it's Job is to select the processes for following

three purposes.

Long-term scheduler(or job scheduler) –selects which processes should be brought into the ready queue

Short-term scheduler(or CPU scheduler) –selects which process should be executed next and allocates CPU.

Mid-term Scheduler (Swapper)- present in all systems with virtual memory, temporarily removes processes from main memory and places them on secondary memory (such as a disk drive) or vice versa. The mid-term scheduler may decide to swap out a process which has not been active for some time, or a process which has a low priority, or a process 28.which is page faulting frequently, or a process which is taking up a large amount of memory in order to free up main memory for other processes, swapping the process back in later when more memory is available, or when the process has been unblocked and is no longer waiting for a resource. Which of the following need not necessarily be saved on a context switch between processes? [B]

(A) General purpose registers **(B)** Translation look aside buffer **(C)** Program counter **(D)** All of the above

28.The following two functions P1 and P2 that share a variable B with an initial value of 2 execute concurrently.

```
P1()
{
  C = B – 1;
  B = 2*C;
}
P2()
{
  D = 2 * B;
  B = D - 1;
}
```

The number of distinct values that B can possibly take after the execution is [A]

(A) 3 **(B)** 2 **(C)** 5 **(D)** 4

Explanation: There are following ways that concurrent processes can follow.

C = B – 1; // C = 1	B = 2*C; // B = 2	D = 2 * B; // D = 4	B = D - 1; // B = 3
C = B – 1; // C = 1	D = 2 * B; // D = 4	B = D - 1; // B = 3	B = 2*C; // B = 2
C = B – 1; // C = 1	D = 2 * B; // D = 4	B = 2*C; // B = 2	B = D - 1; // B = 3
D = 2 * B; // D = 4	C = B – 1; // C = 1	B = 2*C; // B = 2	B = D - 1; // B = 3
D = 2 * B; // D = 4	B = D - 1; // B = 3	C = B – 1; // C = 2	B = 2*C; // B = 4

There are 3 different possible values of B: 2, 3 and 4.

29.Two processes X and Y need to access a critical section. Consider the following synchronization construct used by both the processes.

Process *X*

```
/* other code for process X */
while (true)
{
    varP = true;
    while (varQ = = true)
    {
    /* Critical Section */
        varP = false;
    }
}
/* other code for process X */
```

Process *Y*

```
/* other code for process Y */
while (true)
{
    varQ = true;
    while (varP = = true)
    {
        /* Critical Section */
            varQ = false;
    }
}
/* other code for process Y */
```

Here, varP and varQ are shared variables and both are initialized to false. Which one of the following statements is true? [A]

(A) The proposed solution prevents deadlock but fails to guarantee mutual exclusion
(B) The proposed solution guarantees mutual exclusion but fails to prevent deadlock
(C) The proposed solution guarantees mutual exclusion and prevents deadlock
(D) The proposed solution fails to prevent deadlock and fails to guarantee mutual exclusion
Explanation: When both processes try to enter critical section simultaneously,both are allowed to do so since both shared variables varP and varQ are true.So, clearly there is NO mutual exclusion. Also, deadlock is prevented because mutual exclusion is one of the four conditions to be satisfied for deadlock to happen.Hence, answer is A.

30.In a certain operating system, deadlock prevention is attempted using the following scheme. Each process is assigned a unique timestamp, and is restarted with the same timestamp if killed. Let Ph be the process holding a resource R, Pr be a process requesting for the same resource R, and T(Ph) and T(Pr) be their timestamps respectively. The decision to wait or preempt one of the processes is based on the following algorithm.

if T(Pr) < T(Ph)

then kill Pr

else wait

Which one of the following is TRUE? [A]

(A) The scheme is deadlock-free, but not starvation-free
(B) The scheme is not deadlock-free, but starvation-free
(C) The scheme is neither deadlock-free nor starvation-free
(D) The scheme is both deadlock-free and starvation-free
Explanation:

1. This scheme is making sure that the timestamp of requesting process is always lesser than holding process
2. The process is restarted with same timestamp if killed and that timestamp can NOT be greater than the existing time stamp

From 1 and 2,it is clear that any new process coming having LESSER timestamp will be KILLED.So, **NO DEADLOCK possible**
However, a new process will lower timestamp may **have to wait infinitely** because of its LOWER timestamp(as killed process will also have same timestamp ,as it was killed earlier). **STARVATION IS Definitely POSSIBLE**
So Answer is A
31.A process executes the following segment of code :

```
for(i = 1; i < = n; i++)
```

fork ();

The number of new processes created is [C]

(A) n **(B)** ((n(n + 1))/2) **(C)** $2^n - 1$ **(D)** $3^n - 1$

Explanation:

fork (); // Line 1

fork (); // Line 2

fork (); // Line 3

.....till n

L1 // There will be 1 child process created by line 1

/ \

L2 L2 // There will be 2 child processes created by line 2

/ \ / \

L3 L3 L3 L3 // There will be 4 child processes created by line 3

.....

We can also use direct formula to get the number of child processes.

With n fork statements, there are always $2^n - 1$ child processes.

32.The semaphore variables full, empty and mutex are initialized to 0, n and 1, respectively. Process P_1 repeatedly adds one item at a time to a buffer of size n, and process P_2 repeatedly removes one item at a time from the same buffer using the programs given below. In the programs, K, L, M and N are unspecified statements.

P1

while (1) {

K;

P(mutex);

Add an item to the buffer;

V(mutex);

L;}

P2

while (1) {

M;

P(mutex);

Remove an item from the buffer;

V(mutex);

N;}

The statements K, L, M and N are respectively [D]

(A) P(full), V(empty), P(full), V(empty) **(B)** P(full), V(empty), P(empty), V(full)
(C) P(empty), V(full), P(empty), V(full) **(D)** P(empty), V(full), P(full), V(empty)

Explanation:

Process P1 is the producer and process P2 is the consumer.Semaphore 'full' is initialized to '0'. This means there is no item in the buffer.Semaphore 'empty' is initialized to 'n'. This means there is space for n items in the buffer.In process P1, wait on semaphore 'empty' signifies that if there is no space in buffer then P1 can not produce more items. Signal on semaphore 'full' is to signify that one item has been added to the buffer.In process P2, wait on semaphore 'full' signifies that if the buffer is empty then consumer can't not consume any item. Signal on semaphore 'empty' increments a space in the buffer after consumption of an item.

33.Consider the following two-process synchronization solution.

Process 0 **Process 1**

Entry: loop while (turn == 1); Entry: loop while (turn == 0);
(critical section) (critical section)
Exit: turn = 1; Exit turn = 0;

The shared variable turn is initialized to zero. Which one of the following is TRUE? [C]

(A) This is a correct two-process synchronization solution.
(B) This solution violates mutual exclusion requirement.
(C) This solution violates progress requirement. **(D)** This solution violates bounded wait requirement.

Explanation: A **mutual exclusive** requirement prevents simultaneous access to a shared resource. Since, semaphore turn is initialized to zero and this semaphore value is changing only after critical section for given processes. So, this ensures at most one process in the critical section at a time, i.e., mutual exclusive requirement is satisfied.

Progress means that process should eventually be able to complete. But directly **Process 1** can not go critical section as semaphore value is 0 initially and this process 1 can go CS only after process 0. So, progress requirement is **not** satisfied.

Bounded waiting means no process should wait for a resource for infinite amount of time. Process 1 can go directly and Process 1 can go after process 0 into critical section. So, Bounded waiting is satisfied.

34.Consider a non-negative counting semaphore S. The operation P(S) decrements S, and V(S) increments S. During an execution, 20 P(S) operations and 12 V(S) operations are issued in some order. The largest initial value of S for which at least one P(S) operation will remain blocked is _____. [A]

(A) 7 **(B)** 8 **(C)** 9 **(D)** 10

Explanation: 20-7 -> 13 will be in blocked state, when we perform 12 V(S) operation makes 12 more process to get chance for execution from blocked state. So one process will be left in the queue (blocked state) here i have considered that if a process is in under CS then it not get blocked by other process.

35.In the working-set strategy, which of the following is done by the operating system to prevent thrashing?

1. It initiates another process if there are enough extra frames.
2. It selects a process to suspend if the sum of the sizes of the working-sets exceeds the total number of available frames. [D]

(A) I only **(B)** II only **(C)** Neither I nor II **(D)** Both I and II

Explanation: According to concept of thrashing,

- **I is true** because to prevent thrashing we must provide processes with as many frames as they really need "right now".If there are enough extra frames, another process can be initiated.
- **II is true** because The total demand, D, is the sum of the sizes of the working sets for all processes. If D exceeds the total number of available frames, then at least one process is thrashing, because there are not enough frames available to satisfy its minimum working set. If D is significantly less than the currently available frames, then additional processes can be launched.

36.Processes P_1 and P_2 use critical_flag in the following routine to achieve mutual exclusion. Assume that critical_flag is initialized to FALSE in the main program.

```
get_exclusive_access ( )
{
if (critical _flag == FALSE) {
critical_flag = TRUE ;
critical_region () ;
critical_flag = FALSE;
}
}
```

Consider the following statements.

i. It is possible for both P_1 and P_2 to access critical_region concurrently. ii. This may lead to a deadlock.

Which of the following holds? [C]

(A) (i) is false and (ii) is true **(B)** Both (i) and (ii) are false

(C) (i) is true and (ii) is false **(D)** Both (i) and (ii) are true

```
get_exclusive_access()
{
    if(critical_flage == False){
       critical_flage == True;
       critical_region  ( );
       critical_fflage == False;
    }
 }
```

Explanation:

Say P_1 starts first and executes statement 1, after that system context switches to P_2 (before executing statement 2), and it enters inside if statement, since the flag is still false.So now both processes are in critical section!! so (i) is true.. (ii) is false.By no way it happens that flag is true and no process' are inside the if clause, if someone enters the critical section, it will definitely make flag = false. So no deadlock.

37.The following is a code with two threads, producer and consumer, that can run in parallel. Further, S and Q are binary semaphores equipped with the standard P and V operations.

```
semaphore S = 1, Q = 0;
integer x;
producer:                consumer:
while (true) do          while (true) do
P(S);                    P(Q);
x = produce ();          consume (x);
V(Q);                    V(S);
done                     done
```

Which of the following is TRUE about the program above? [D]

(A) The process can deadlock **(B)** One of the threads can starve

(C) Some of the items produced by the producer may be lost

(D) Values generated and stored in 'x' by the producer will always be consumed before the producer can generate a new value

Explanation: A semaphore is hardware or a software tag variable whose value indicates the status of a common resource. Its purpose is to lock the resource being used. A process which needs the resource will check the semaphore for determining the status of the resource followed by the decision for proceeding. In multitasking operating systems, the activities are synchronized by using the semaphore techniques.

wait and signal are defined on the semaphore. Entry to the critical section is controlled by the wait operation and exit from a critical region is taken care by signal operation. The wait, signal operations are also called P and V operations. The manipulation of semaphore (S) takes place as following:

1. The wait command P(S) decrements the semaphore value by 1. If the resulting value becomes negative then P command is delayed until the condition is satisfied.
2. The V(S) i.e. signals operation increments the semaphore value by 1.Consumer can consume only once the

producer has produced the item, and producer can produce(except the first time) only once the consumer has consumed the item.Let's explain the working of this code.It is mentioned that Producer and Consumer execute parallely.

Producer:st1 – S value is 1, P(S) on S makes it 0 and
st2 – and then x item is produced.
st3 – Q value is 0. V(Q) on Q makes it 1.this being an infinite while loop should infinitely iterate. In the next iteration of while loop, S is already 0 ,further P(S) on 0 sends P to blocked list of S. So producer is blocked.

Consumer:

P(Q) on Q makes Q =0 and then consumes the item.
V(S) on S,
now instead of changing the value of S to 1,consumer wakes up the blocked process on Q 's queue. Hence process P is awaken. P resumes from st2,since it was blocked at statement 1. So P now produces the next item. So consumer consumes an item before producer produces the next item.
Choice of this question:A) deadlock cannot happen has both producer and consumer are operating on different semaphores (no hold and wait)
B) No starvation happen because there is alteration between P and Consumer.Which also makes them have bounded waiting. (C) Some of the items produced by the producer may be lost but it can't.

39.An operating system implements a policy that requires a process to release all resources before making a request for another resource. Select the TRUE statement from the following: [B]
(A) Both starvation and deadlock can occur **(B)** Starvation can occur but deadlock cannot occur
(C) Starvation cannot occur but deadlock can occur **(D)** Neither starvation nor deadlock can occur
Explanation: Starvation may occur, as a process may want othe resource in ||^{al} along with currently hold resources.

According to given conditions it will never be possible to collect all at a time.

No deadlock.

40.If the time-slice used in the round-robin scheduling policy is more than the maximum time required to execute any process, then the policy will [C]

(A) degenerate to shortest job first **(B)** degenerate to priority scheduling
(C) degenerate to first come first serve **(D)** none of the above
Explanation: RR executes processes in FCFS manner with a time slice. It this time slice becomes long enough, so that a process finishes within it, It becomes FCFS.
41Consider the following C code for process P1 and P2. a=4, b=0, c=0 (initialization)

```
 P1                P2
if (a < 0)         b = 10;
  c = b-a;         a = -3;
else
  c = b+a;
```

If the processes P1 and P2 executes concurrently (shared variables a, b and c), which of the following cannot be the value of 'c' after both processes complete? [D]

(A) 4 **(B)** 7 **(C)** 10 **(D)** 13
Explanation:
P1 : 1, 3, 4 -> c = 0+4 =4 {hence option a}
P2 : i, ii and P1 : 1, 2 -> c = 10-(-3) = 13 {hence option d}
P1 : 1 , P2 : i, ii and P1 : 3, 4 -> c= 10+(-3) = 7 { hence option b}

So 10 cannot be c value.

42.Which of the following actions is/are typically not performed by the operating system when switching context from process A to process B? [C]

(A) Saving current register values and restoring saved register values for process B.
(B) Changing address translation tables. **(C)** Swapping out the memory image of process A to the disk.
(D) Invalidating the translation look-aside buffer.

Explanation: During context switch processes are not swapped out from memory to disk but processes are generally swapped out from memory to disk when they are suspended. Also, during context switch OS invalidates TLB so that the TLB coincides with the currently executing processes.

43.In a computer system where the 'best-fit' algorithm is used for allocating 'jobs' to 'memory partitions', the following situation was encountered:

Partitions size in KB	4K 8K 20K 2K
Job sizes in KB	2K 14K 3K 6K 6K 10K 20K 2K
Time for execution	4 10 2 1 4 1 8 6

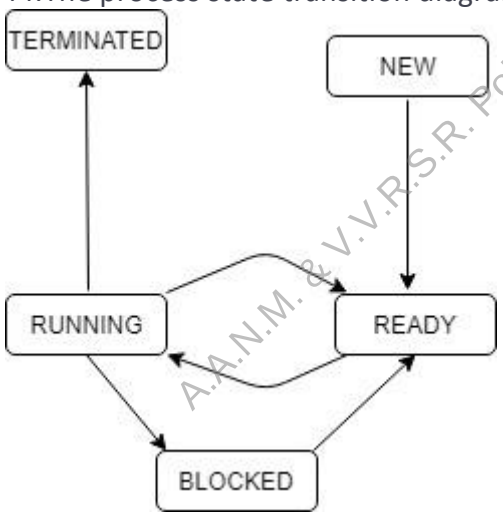
When will the 20K job complete? [D]

Note – This question was subjective type.

(A) 8 **(B)** 18 **(C)** 11 **(D)** None of these

Explanation: Partitions are 4k , 8k, 20k, 2k, now due to best fit allocation algorithm.
Best fit algo allocates the process to a partition which is the smallest sufficient partition among the free available partitions now come to the question first 2k come and fit into the partition of 2k and run (4 unit) and then come 14k and fit into the partition of 20k and run(10 unit) and next process come of size 3k and fit into the portion 4k and run (2 unit) and now next process 6k come and fit to the 8k partition and run(1 unit) now next process is 10k and fit into portion20k run for (1 unit) and finally 20k come and fit into 20k portion and run for (8 unit).So total completion time for 20k job = (10+1+8) = **19 unit** answer.

44.The process state transition diagram in below figure is representative of [B]



(A) a batch operating system **(B)** an operating system with a preemptive schedular
(C) an operating system with a non-preemptive scheduler **(D)** a uni-programmed operating system

45.Which of the following option is False? [B]

(A) An executing instance of a program is called a process while a thread is a subset of the process.
(B) Threads have considerable overhead while processes have almost no overhead.
(C) Execution of processes are independent while execution of threads are dependent.
(D) New processes require duplication of the parent process while new threads are easily created.

Explanation: Processes have considerable overhead while threads have almost no overhead.

46.Which of the following is/are not shared by all the threads in a process? [C]

I. Program Counter II. Stack III. Registers IV. Address space

(A) I and II only **(B)** II and III only **(C)** I, II and III only **(D)** IV only

Explanation: Every thread have its own stack, register, and PC, so only address space that is shared by all thread for a single process.

Option (C) is correct.

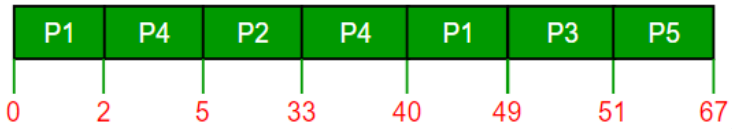
47.Consider the set of processes with arrival time(in milliseconds), CPU burst time (in milliseconds), and priority(0 is the highest priority) shown below. None of the processes have I/O burst time.

Process	Arrival Time	Burst Time	Priority
P1	0	11	2
P2	5	28	0
P3	12	2	3
P4	2	10	1
P5	9	16	4

The waiting time (in milliseconds) of process P1 using preemptive priority scheduling algorithm is [C]

(A) 26 **(B)** 49 **(C)** 38 **(D)** 29

Explanation: The gantt chart is using preemptive priority scheduling algorithm:



Waiting Time = Completion time – Arrival time – Burst Time

Therefore, waiting time of process P1 is = 49 – 0 – 11 = 38

48.The following are some events that occur after a device controller issues an interrupt while process L is under execution.

(P) The processor pushes the process status of L onto the control stack.
(Q) The processor finishes the execution of the current instruction.
(R) The processor executes the interrupt service routine.
(S) The processor pops the process status of L from the control stack.
(T) The processor loads the new PC value based on the interrupt.

Which of the following is the correct order in the which the events above occur? [A]

(A) QPTRS **(B)** PTRSQ **(C)** TRPQS **(D)** QTPRS

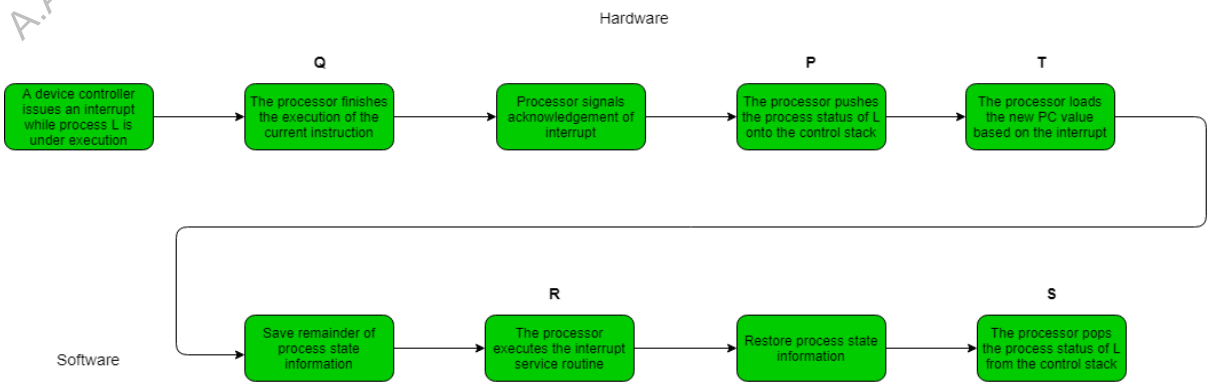
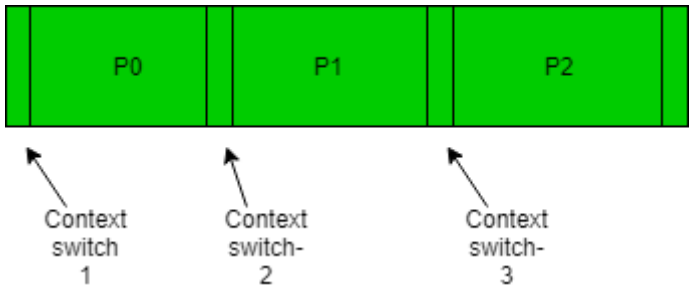


Figure - Simple Interrupt Processing

Explanation:

49.Two atomic operations permissible on Semaphores are _____ and _____. [D]
(A) wait, stop **(B)** wait, hold **(C)** hold, signal **(D)** wait, signal
50.Consider three CPU intensive processes, which require 10, 20, 30 units and arrive at times 0, 2, 6 respectively. How many context switches are needed if shortest remaining time first is implemented? Context switch at 0 is included but context switch at end is ignored [C]
(A) 1 **(B)** 2 **(C)** 3 **(D)** 4
Explanation: Let three process be P0, P1 and P2 with arrival times 0, 2 and 6 respectively and CPU burst times 10, 20 and 30 respectively. At time 0, P0 is the only available process so it runs. At time 2, P1 arrives, but P0 has the shortest remaining time, so it continues. At time 6, P2 also arrives, but P0 still has the shortest remaining time, so it continues. At time 10, P1 is scheduled as it is the shortest remaining time process. At time 30, P2 is scheduled.



51.A process executes the following code

for (i = 0; i < n; i++) fork( );

The total number of child processes created are [D]
(A) n^2 **(B)** 2^{n+1} -1 **(C)** 2^n **(D)** 2^n -1
52.Which of the following is false? [D]
(A) User level threads are not scheduled by the kernel
(B) Context switching between user level threads is faster than context switching between kernel level threads
(C) When a user thread is blocked all other threads of its processes are blocked
(D) Kernel level threads cannot utilize multiprocessor systems by splitting threads on different processors or cores
53.User level threads are threads that are visible to the programmer and are unknown to the kernel. The operating system kernel supports and manages kernel level threads. Three different types of models relate user and kernel level threads.
Which of the following statements is/are true ? [A]
(a)
(i) The Many – to – one model maps many user threads to one kernel thread
(ii) The one – to – one model maps one user thread to one kernel thread
(iii) The many – to – many model maps many user threads to smaller or equal kernel threads
(b)
(i) Many – to – one model maps many kernel threads to one user thread
(ii) One – to – one model maps one kernel thread to one user thread
(iii) Many – to – many model maps many kernel threads to smaller or equal user threads
(A) (a) is true; (b) is false **(B)** (a) is false; (b) is true
(C) Both (a) and (b) are true **(D)** Both (a) and (b) are false
Explanation: (a)
(i) The Many – to – one model maps many user threads to one kernel thread
(ii) The one – to – one model maps one user thread to one kernel thread
(iii) The many – to – many model maps many user threads to smaller or equal kernel threads
(b)
(i) Many – to – one model maps many kernel threads to one user thread
(ii) One – to – one model maps one kernel thread to one user thread

(iii) Many – to – many model maps many kernel threads to smaller or equal user threads
All satement are correct.

54.The Bounded buffer problem is also known as _____. [A]

(A) Producer – consumer problem **(B)** Reader – writer problem
(C) Dining Philosophers problem **(D)** Both (2) and (3)

Explanation: The bounded buffer problem is also known as producer-consumer problem.

55.What is the output of the following program? [A]

```
main( )
{
int a = 10;
if ((fork ( ) == 0))
a++;
printf ("%d\n", a );
}
```

(A) 10 and 11 **(B)** 10 **(C)** 11 **(D)** 11 and 11

56.Mutual exclusion problem occurs [B]

(A) Between two disjoint processes that do not interact **(B)** Among processes that share resources
(C) Among processes that do not use the same resource
(D) Between two processes that uses different resources of different machine

57.A critical region [A]

(A) is a piece of code which only one process executes at a time **(B)** is a region prone to deadlock
(C) is a piece of code which only a finite number of processes execute
(D) is found only in windows NT operating system

58.At particular time, the value of a counting semaphore is 10, it will become 7 after:
(a) 3 V operations (b) 3 P operations (c) 5 V operations and 2 P operations (d) 2 V operations and 5 P operations
Which of the following option is correct? [C]

(A) Only (b) **(B)** Only(d) **(C)** Both (b) and (d) **(D)** None of these

Explanation: P: Wait operation decrements the value of the counting semaphore by 1.
V: Signal operation increments the value of counting semaphore by 1.
Current value of the counting semaphore = 10
a) after 3 P operations, value of semaphore = 10-3 = 7
d) after 2 v operations, and 5 operations value of semaphore = 10 + 2 – 5 = 7

59.There are three processes P_1, P_2 and P_3 sharing a semaphore for synchronizing a variable. Initial value of semaphore is one. Assume that negative value of semaphore tells us how many processes are waiting in queue. Processes access the semaphore in following order :
(a) P_2 needs to access (b) P_1 needs to access (c) P_3 needs to access (d) P_2 exits critical section
(e) P_1 exits critical section
The final value of semaphore will be : [A]

(A) 0 **(B)** 1 **(C)** -1 **(D)** -2

Explanation: Initial value of semaphores S=1

1. P_2 needs to access decreases semaphore by 1, new value will be 0 (no one is waiting)
2. P_1 needs to access decreases semaphore by 1, new value will be -1 (one process is waiting)
3. P_3 needs to access decreases semaphore by 1, new value will be -2 (2 process are waiting)
4. P_2 exits critical section increases semaphore by 1, new value will be -1 (one process is waiting)
5. P_1 exits critical section increases semaphore by 1, new value will be 0 (no process is waiting)

60.At a particular time of computation the value of a counting semaphore is 7. Then 20 P operations and xV operations were completed on this semaphore. If the new value of semaphore is 5 ,x will be [A]

(A) 18 **(B)** 22 **(C)** 15 **(D)** 13

Explanation: P operation : Decrements the value of semaphore by 1
V operation : Increments the value of semaphore by 1
Initially, value of semaphore = 7
After 20 P operations, value of semaphore = 7 – 20 = -13
Now, after xV operations, value of semaphore = 5
-13 + xV = 5
xV = 5 + 13 = 18
So, option (A) is correct.

61.One of the disadvantages of user level threads compared to Kernel level threads is [A]

(A) If a user-level thread of a process executes a system call, all threads in that process are blocked.
(B) Scheduling is application dependent. **(C)** Thread switching doesn't require kernel mode privileges.
(D) The library procedures invoked for thread management in user level threads are local procedures.

Explanation: Advantage of User level thread:
1- Scheduling is application dependent.
2- Thread switching doesn't require kernel mode privileges.
3- The library procedures invoked for thread management in user level threads are local procedures.
4- User level threads are fast to create and manage.
5- User level thread can run on any operating system.
Disadvantage of User-level thread
1- Most system calls are blocked on a typical OS.
2- Multiprocessing is not supported for multi-threaded application.

62.Names of some of the Operating Systems are given below:
(a) MS-DOS
(b) XENIX
(c) OS/2
In the above list, following operating systems didn't provide multiuser facility. [D]

(A) (a) only **(B)** (a) and (b) only **(C)** (b) and (c) only **(D)** (a), (b) and (c)

Explanation:MS-DOS is an operating system for x86-based personal computers mostly developed by Microsoft. It doesn't provide multi-user facility.**XENIX** is a discontinued version of the Unix operating system for various microcomputer platforms,licensed by Microsoft from AT&T Corporation. It doesn't provide multi-user facility.**OS/2** is a series of computer operating systems, initially created by Microsoft and IBM. It doesn't provide multi-user facility.

63.In a lottery scheduler with 40 tickets, how we will distribute the tickets among 4 processes and such that each process gets 10%, 5%, 60% and 25% respectively? [C]

	P1	P2	P3	P4
a)	12	4	70	30
b)	7	5	20	10
c)	4	2	24	10
d)	8	5	40	30

(A) a **(B)** b **(C)** c **(D)** d

Explanation: Total tickets = 40
4 processes to get 10%, 5%, 60% and 25%
10% of 40 = 4; 5% of 40 = 2;
60% of 40 = 24 and 25% of 40 = 10

64.Suppose a system contains n processes and system uses the round-robin algorithm for CPU scheduling then which data structure is best suited for ready queue of the process. [C]

(A) stack **(B)** queue **(C)** circular queue **(D)** tree

65.Consider a system with seven processes A through G and six resources R through W.
Resource ownership is as follows :

process A holds R and wants T process B holds nothing but wants T
process C holds nothing but wants S process D holds U and wants S & T
process E holds T and wants V process F holds W and wants S
process G holds V and wants U
Is the system deadlocked ? If yes, _____ processes are deadlocked. [C]
(A) No **(B)** Yes, A, B, C **(C)** Yes, D, E, G **(D)** Yes, A, B, F

66.Consider a system having 'm' resources of the same type. These resources are shared by three processes P_1, $P_2$and P_3 which have peak demands of 2, 5 and 7 resources respectively. For what value of 'm' deadlock will not occur ? [B]
(A) 70 **(B)** 14 **(C)** 13 **(D)** 7
Explanation: To avoid deadlock 'm' >= peak demands($P_1 + P_2 + P_3$)
i.e. m >= peak demands(2 + 5 + 7)
m >= peak demands(14)

67.An Operating System (OS) crashes on the average once in 30 days, that is, the Mean Time Between Failures (MTBF) = 30 days. When this happens, it takes 10 minutes to recover the OS, that is, the Mean Time To Repair (MTTR) = 10 minutes. The availability of the OS with these reliability figures is approximately : [D]
(A) 96.97% **(B)** 97.97% **(C)** 99.009% **(D)** 99.97%
Explanation: System crashes once in 30 days and need 10 minutes to get repaired.
Either convert 30 days into minute or 10 minute into days
30 days = 30 ∗ 24 ∗ 60 minute
fraction of time system is not available = (10 / (30 ∗ 24 ∗ 60)) ∗ 100 = 0.023%
Availability = 100 – 0.023 = 99.97%

68.Suppose there are four processes in execution with 12 instances of a Resource R in a system.
The maximum need of each process and current allocation are given below:

Process	Max. need	Allocation
P1	8	3
P2	9	4
P3	5	2
P4	3	1

With reference to current allocation, is system safe ? If so, what is the safe sequence ? [C]
(A) No **(B)** Yes, $P_1P_2P_3P_4$ **(C)** Yes, $P_4P_3P_1P_2$ **(D)** Yes, $P_2P_1P_3P_4$
Explanation: Current allocation of $P_1P_2P_3P_4$ are 3, 4, 2, 1 which is 10 in total.
We have 12 total no of resources and out of them 10 are allocated so, we have only 2 resources.
There is 5, 5, 3, 2 resources are needed for $P_1P_2P_3P_4$ respectively.
So, P_4 will run first and free 3 resources after execution. Which are sufficient for P_3 So it will execute and do free 5 resources.
Now P_1 and P_2 both require 5 resources each So we can execute any of them first but we will give priority to P_1.
The execution order will be $P_4P_3P_1P_2$..SO, option (C) is correct.

69.Consider the following set of processes, with arrival times and the required CPU-burst times given in milliseconds.

Process	**Arrival Time**	**Burst Time**
P1	0	4
P2	2	2

P3 3 1

What is the sequence in which the processes are completed? Assume round robin scheduling with a time quantum of 2 milliseconds. [B]

(A) P1, P2, P3 **(B)** P2, P1, P3 **(C)** P3, P2, P1 **(D)** P2, P3, P1

Explanation: The sequence in which the processes complete with round-robin algorithm with tq = 2 ms is P2, P1, P3

70.A CPU scheduling algorithm determines an order for the execution of its scheduled processes. Given 'n' processes to be scheduled on one processor, how many possible different schedules are there? [C]

(A) n **(B)** n^2 **(C)** n! **(D)** 2^n

Explanation: For 'n' processes to be scheduled on one processor, there can be n! different schedules possible.Example: Suppose an OS has 4 processes to schedule P1, P2, P3 and P4. For scheduling the first process, it has 4 choices, then from the remaining three processes it can take 3 choices, and so on. So, total schedules possible are 4*3*2*1 = 4!

71.A starvation free job scheduling policy guarantees that no job indefinitely waits for a service. Which of the following job scheduling policies is starvation free? [D]

(A) Priority queing **(B)** Shortest Job First **(C)** Youngest Job First **(D)** Round robin

Explanation: Round Robin is a starvation free scheduling algorithm as it imposes a strict time bound on the response time of each process i.e. for a system with 'n' processes running in a round robin system with time quanta t_q, no process will wait for more than (n-1) t_q time units to get its CPU turn.

72.The state of a process after it encounters an I/O instruction is [B]

(A) ready **(B)** blocked **(C)** idle **(D)** running

Explanation: Whenever a process is just created, it is kept in **Ready** queue. When it starts execution, it is in **Running** state, as soon as it starts doing input/output operation, it is kept in the **blocked** state.

73.Which of the following strategy is employed for overcoming the priority inversion problem? [A]

(A) Temporarily raise the priority of lower priority level process **(B)** Have a fixed priority level scheme

(C) Implement kernel pre-emption scheme **(D)** Allow lower priority process to complete its job

Explanation: Priority inversion is a scenario in scheduling when a higher priority process is indirectly preempted by a lower priority process, thereby inverting the relative priorities of the process. This problem can be eliminated by temporarily raising the priority of lower priority level process, so that it can not preempt the higher priority process.

74.System calls are usually invoked by using : [C]

(A) A privileged instruction **(B)** An indirect jump **(C)** A software interrupt **(D)** Polling

Explanation:

- System calls are usually invoked by using a software interrupt.
- Polling is the process where the computer or controlling device waits for an external device to check for its readiness or state, often with low-level hardware.
- Privileged instruction is an instruction (usually in machine code) that can be executed only by the operating system in a specific mode.
- In direct jump, the target address (i.e. its relative offset value) is encoded into the jump instruction itself.

75.The following table shows the processes in the ready queue and time required for each process for completing its job.

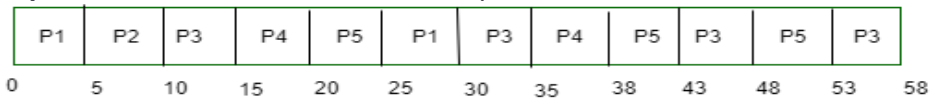
Process	Time
P1	10
P2	5

P3 20
P4 8
P5 15

If round-robin scheduling with 5 ms is used what is the average waiting time of the processes in the queue?[B]

(A) 27 ms **(B)** 26.2 ms **(C)** 27.5 ms **(D)** 27.2 ms

Explanation: The Gantt chart for the processes is shown:



Waiting time of a process = Sum of the periods spent waiting in the ready queue.

Waiting time = completion time - burst time

Waiting time of P1 = 30 -10 = 20 Waiting time of P2 = 10 - 5 = 5 Waiting time of P3 = 58 - 20 = 38

Waiting time of P4 = 38 - 8 = 30 Waiting time of P5 = 53 - 15 = 38

Average waiting time = 20 + 5 + 38 + 30 + 38 =131/5 = 26.2

76.Consider a system with twelve magnetic tape drives and three processes P_1, P_2 and P_3. Process P_1 requires maximum ten tape drives, process P_2 may need as many as four tape drives and P_3 may need upto nine tape drives. Suppose that at time t_1, process P_1 is holding five tape drives, process P_2 is holding two tape drives and process P_3 is holding three tape drives. At time t_1, system is in: [B]

(A) safe state **(B)** unsafe state **(C)** deadlocked state **(D)** starvation state

Explanation: P_1 is holding 5 tape drive but it require 10 tape drive, 5 tapes are required for P_1.P_2 is holding 2 tape drive but it require 4 tape drive, 2 tapes are required for P_2.P_3 is holding 3 tape drive but it require 9 tape drive, 6 tapes are required for P_3.We have total 12 tape out of which 10 are allocated, 2 tape are free. These 2 tape can be allocated to P_2. After P_2 complete its execution it will free 4 tape, but these 4 tape can not fulfill requirement for P_1 or P_3. That's why system is in unsafe state.

77.In an operating system, indivisibility of operation means: [C]

(A) Operation is interruptable **(B)** Race – condition may occur

(C) Processor can not be pre-empted **(D)** All of the above

Explanation: In an operating system, indivisibility of operation means processor can not be pre-empted. One a process starts its execution it will not suspended or stop its execution inside the processor.

78.There are three processes in the ready queue. When the currently running process requests for I/O how many process switches take place? [A]

(A) 1 **(B)** 2 **(C)** 3 **(D)** 4

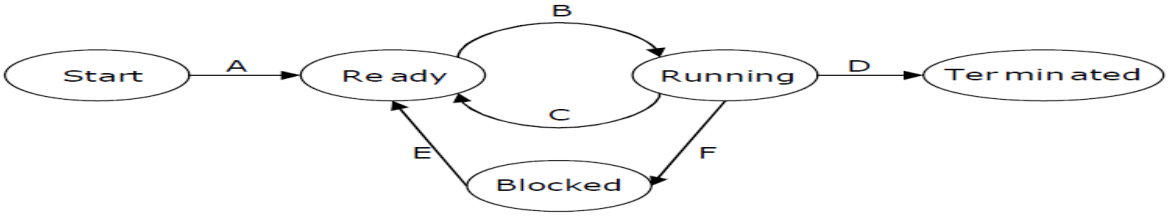
Explanation: Single process switch will take place as when the currently running process requests for I/O, it would be placed in the blocked list and the first process residing inside the Ready Queue will be placed in the Running list to start its execution.

79.Which is the correct definition of a valid process transition in an operating system? [B]

(A) Wake up: ready → running **(B)** Dispatch: ready → running

(C) Block: ready → running **(D)** Timer runout: ready → running

Explanation: The state transition diagram of a process(preemptive scheduling):



Option 1: Wake up: ready → running

It is incorrect as when a process wakes up it is shifted from blocked state to ready state and not from ready to running.

Option 2: Dispatch: ready → running
It is correct as the dispatcher selectively assigns the CPU to one of the process in the ready queue based on a well defined algorithm.
Option 3: Block: ready → running
It is incorrect as a process is blocked when it is either pre-empted by some other process or due to some i/o operation. So when a process gets blocked it shifts from running state to blocked state.
Option 4: Timer runout: ready → running
When the time duration of execution of a process expires, the timer interrupts and the process shifts from the running state to ready queue.

80.The correct matching of the following pairs is [D]

(A) Disk check (1) Roundrobin

(B) Batch processing (2) Scan

(C) Time sharing (3) LIFO

(D) Stack operation (4) FIFO

(A)
A B C D

3 4 2 1

(B)

A B C D

4 3 2 1

(C)
A B C D

3 4 1 2

(D)
A B C D

2 4 1 3

Explanation: Option (D) is correct:
A: Disk check- (2) Various scan algorithms are used for disc checking like FCFS, SSTF, SCAN, C-SCAN, LOOK, C-LOOK etc.
B: Batch Processing- In batch processing processes are strictly executed in FIFO order.
C: Time sharing- In Round Robin scheduling algorithm, the processes perform CPU execution in time sharing manner as there is a fixed time quanta associated with each process for execution.
D: Stack operation- stack operates in last in first out manner.

81.Consider three CPU-intensive processes, which require 10, 20 and 30 time units and arrive at times 0, 2 and 6, respectively. How many context switches are needed if the operating system implements a shortest remaining time first scheduling algorithm? [B]
Do not count the context switches at time zero and at the end.
(A) 1 **(B)** 2 **(C)** 3 **(D)** 4
82. The performance of Round Robin algorithm depends heavily on
(A) size of the process **(B)** the I/O bursts of the process
(C) the CPU bursts of the process **(D)** the size of the time quantum
Explanation: In round robin algorithm, the size of time quanta plays a very important role as:If size of quanta is too small: Context switches will increase and it is counted as the waste time, so CPU utilization will

decrease.If size of quanta is too large: Larger time quanta will lead to Round robin regenerated into FCFS scheduling algorithm.

83.Consider a set of 5 processes whose arrival time, CPU time needed and the priority are given below:

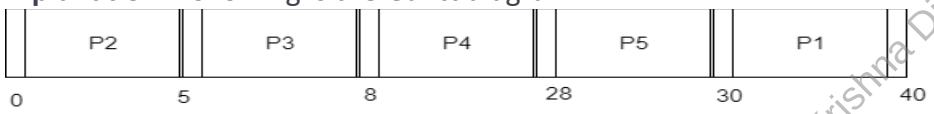
Process	Arrival Time (in ms)	CPU Time Needed	Priority
P1	0	10	5
P2	0	5	2
P3	2	3	1
P4	5	20	4
P5	10	2	3

(smaller the number, higher the priority)

If the CPU scheduling policy is priority scheduling without preemption, the average waiting time will be [C]

(A) 12.8 ms **(B)** 11.8 ms **(C)** 10.8 ms **(D)** 9.8 ms

Explanation: Following is the Gantt diagram:



Waiting time = turn around time – burst time

Turn around time = completion time – arrival time

	Arrival Time	CPU Time Needed	Priority	turnaround time	Waiting Time(in ms)
P1	0	10	5	40 - 0 = 40	40 - 10 = 30
P2	0	5	2	5 - 0 = 5	5 - 5 = 0
P3	2	3	1	8 - 2 = 6	6 - 3 = 3
P4	5	20	4	28 - 5 = 23	23 -20 = 3
P5	10	2	3	30 - 10 = 20	20 - 2 = 18

Average Waiting Time = (30 + 3 + 3 + 18)/ 5 = 10.8

84.Four jobs to be executed on a single processor system arrive at time 0 in the order A, B, C, D . Their burst CPU time requirements are 4, 1, 8, 1 time units respectively. The completion time of A under round robin scheduling with time slice of one time unit is [D]

(A) 10 **(B)** 4 **(C)** 8 **(D)** 9

Explanation: The order of execution of the processes with the arrival time of each process = 0, using round robin algorithm with time quanta = 1

A B C D A C A C A, i.e, after 8 context switches, A finally completes it execution

85.Which of the following need not necessarily be saved on a context switch between processes? [B]

(A) General purpose registers **(B)** Translation look-aside buffer

(C) Program counter **(D)** All of the above

Explanation: The values stored in registers, stack pointers and program counters are saved on context switch between the processes so as to resume the execution of the process.

There's no need of saving the contents of TLB as it is invalidated after each context switch.

86.With Round-Robin CPU scheduling in a time shared system [A]

(A) using very large time slices (quantas) degenerates into FirstCome First served (FCFS) algorithm.

(B) using extremely small time slices improves performance
(C) using very small time slices degenerates into Last-In First-Out (LIFO) algorithm.
(D) using medium sized times slices leads to shortest Request time First (SRTF) algorithm
Explanation: If time quantum is very large, then scheduling happens according to FCFS.
87.What is the name of the operating system that reads and reacts in terms of operating system? [C]
(A) Batch system **(B)** Quick response time
(C) real time system **(D)** Time sharing system
Explanation: A real-time operating system is an operating system that guarantees to process events or data by a specific moment in time.
88.Fork is [D]
(A) the creation of a new job **(B)** the dispatching of a task
(C) increasing the priority of a task **(D)** the creation of a new process
Explanation: fork() creates a new process by duplicating the calling process, The new process, referred to as child, is an exact duplicate of the calling process, referred to as parent, except for the following :The child has its own unique process ID, and this PID does not match the ID of any existing process group.The child's parent process ID is the same as the parent's process ID.The child does not inherit its parent's memory locks and semaphore adjustments.The child does not inherit outstanding asynchronous I/O operations from its parent nor does it inherit any asynchronous I/O contexts from its parent.
89.Which of the following need not necessarily be saved on a Context Switch between processes? [B]
(A) General purpose registers **(B)** Translation look-aside buffer
(C) Program counter **(D)** Stack pointer
Explanation: The values stored in registers, stack pointers and program counters are saved on context switch between the processes so as to resume the execution of the process.There's no need for saving the contents of TLB as it is being invalid after each context switch.
90.The performance of Round Robin algorithm depends heavily on [D]
(A) size of the process **(B)** the I/O bursts of the process
(C) the CPU bursts of the process **(D)** the size of the time quantum
Explanation: In Round Robin algorithm, it is very important to choose the the quantum carefully as smaller time quanta leads to more context switches thereby reducing the efficiency of the CPU and the larger time quanta makes the round robin algorithm regenerate into FCFS algorithm.
91.Match the following: [B]

List – I	List – II
(a) Spooling	(i) Allows several jobs in memory to improve CPU utilization
(b) Multiprogramming	(ii)Access to shared resources among geographically dispersed computers in a transparent way
(c)Time sharing	(iii) Overlapping I/O and computations
(d)Distributed computing	(iv)Allows many users to share a computer simultaneously by switching processor frequently

codes:

	(a)	(b)	(c)	(d)
(1)	(iii)	(i)	(ii)	(iv)

(2) (iii) (i) (iv) (ii)

(3) (iv) (iii) (ii) (i)

(4) (ii) (iii) (iv) (i)

(A) (1) **(B)** (2) **(C)** (3) **(D)** (4)

Explanation:

- Spooling provides Overlapping I/O and computations.
- Multiprogramming allows several jobs in memory to improve CPU utilization.
- Time sharing allows many users to share a computer simultaneously by switching processor frequently.
- Distributed computing accesses to shared resources among geographically dispersed computers in a transparent way.

92.Dining Philosopher's problem is a: [B]

(A) Producer – consumer problem **(B)** Classical IPC problem

(C) Starvation problem **(D)** Synchronization primitive

93.Consider a set of n tasks with known runtimes r_1, r_2….r_n to be run on a uniprocessor machine. Which of the following processor scheduling algorithms will result in the maximum throughput? [B]

(A) Round Robin **(B)** Shortest job first

(C) Highest response ratio next **(D)** first come first served

Explanation: Throughput means total number of tasks executed per unit time i.e. sum of waiting time and burst time.

Shortest job first scheduling is a scheduling policy that selects the waiting process with the smallest execution time to execute next.

Thus, in shortest job first scheduling, shortest jobs are executed first. This means CPU utilization is maximum. So, maximum number of tasks are completed.

94.Round Robin schedule is essentially the pre-emptive version of [A]

(A) FIFO **(B)** Shortest job first

(C) Shortest remaining time **(D)** Longest remaining time

Explanation: FIFO when implemented in preemptive version, acts like a round-robin algorithm.

95.What is the name of the technique in which the operating system of a computer executes several programs concurrently by switching back and forth between them? [B]

(A) Partitioning **(B)** Multi-tasking **(C)** Windowing **(D)** Paging

Explanation: In a multitasking system, a computer executes several programs simultaneously by switching them back and forth to increase the user interactivity. Processes share the CPU and execute in an interleaving manner. This allows the user to run more than one program at a time.

96.A task in a blocked state [D]

(A) is executable **(B)** is running **(C)** must still be placed in the run queues

(D) is waiting for some temporarily unavailable resources

Explanation: Waiting or Blocked state is when a process has requested some input/output and is waiting for the resource.

97.On a system using non-preemptive scheduling, processes with expected run times of 5, 18, 9 and 12 are in the ready queue. In what order should they be run to minimize wait time? [B]

(A) 5, 12, 9, 18 **(B)** 5, 9, 12, 18**(** **C)** 12, 18, 9, 5 **(D)** 9, 12, 18, 5

Explanation: The processes should execute in SJF manner to get the lowest waiting time. So, the order should be 5, 9, 12, 18.

98.Feedback queues [B]

(A) are very simple to implement **(B)** dispatch tasks according to execution characteristics

(C) are used to favour real time tasks **(D)** require manual intervention to implement properly

Explanation: Multilevel Feedback Queue Scheduling (MLFQ) keep analyzing the behavior (time of execution) of processes and according to which it changes its priority of execution of processes.

99.Which of the following conditions does not hold good for a solution to a critical section problem ? [C]

(A) No assumptions may be made about speeds or the number of CPUs.

(B) No two processes may be simultaneously inside their critical sections.

(C) Processes running outside its critical section may block other processes.

(D) Processes do not wait forever to enter its critical section.

Explanation: In Critical section problem:

- No assumptions may be made about speeds or the number of CPUs.
- No two processes may be simultaneously inside their critical sections.
- Processes running outside its critical section can't block other processes getting enter into critical section.
- Processes do not wait forever to enter its critical section.

100.For switching from a CPU user mode to the supervisor mode following type of interrupt is most appropriate [C]

(A) Internal interrupts **(B)** External interrupts **(C)** Software interrupts **(D)** None of the above

Explanation: For switching from a CPU user mode to the supervisor mode Software interrupts occurs. Software interrupts is internal interrupt triggered by some some software instruction. And external interrupt is caused by some hardware module.

101.Monitor is an Interprocess Communication (IPC) technique which can be described as [A]

(A) It is higher level synchronization primitive and is a collection of procedures, variables, and data structures grouped together in a special package.

(B) It is a non-negative integer which apart from initialization can be acted upon by wait and signal operations.

(C) It uses two primitives, send and receive which are system calls rather than language constructs.

(D) It consists of the IPC primitives implemented as system calls to block the process when they are not allowed to enter critical region to save CPU time.

Explanation: Monitor is an Interprocess Communication (IPC) technique which can be described as – higher level synchronization primitive and is a collection of procedures, variables, and data structures grouped together in a special package.

102.The following C program

```
main()
{
 fork() ; fork() ; printf ("yes");
}
```

If we execute this core segment, how many times the string yes will be printed ? [C]

(A) Only once **(B)** 2 times **(C)** 4 times **(D)** 8 times

Explanation: Number of times YES printed is equal to number of process created. Total Number of Processes = 2^n where n is number of fork system calls. So here n = 2, $2^4 = 4$

```
fork ();   // Line 1
fork ();   // Line 2
```

```
     P1
   /   \
  P1    C1
 / \   / \
P1  C2  C1  C3
```

So, there are total 4 processes (3 new child processes and one original process).

103.Here are the two concurrent process P_1, P_2 with respective codes:

P_1 code:

```
while (true) // infinite condition
{
A :____;
printf("%d", 1);
printf("%d", 1);
B:____;
}
```

P_2 code:

```
while (true) // infinite condition
{
C:____;
printf("%d", 0);
printf("%d", 0);
D:____;
}
```

What should be the binary semaphore operation on A,B,C,D respectively and what must be the intial values of semaphore M,N inorder to get the output 110011001100….?
Where P is down and V is up operation respectively. [A]
(A) A = P(N), B = V(M), C = P(M), D = V(N); M = 0, N = 1; **(B)** A = P(N), B = V(M), C = P(M), D = P(N); M = N = 1;
(C) A = P(N), B = V(N), C = P(M), D = V(M); M = 1, N = 0; **(D)** A = P(N), B = V(N), C = P(M), D = V(M); M = N = 1;

104.At a particular time of computation, the value of a counting semaphore is 10. Then 12 P operations and "x" V operations were performed on this semaphore. If the final value of semaphore is 7, x will be: [B]
(A) 8 **(B)** 9 **(C)** 10 **(D)** 11

Explanation: Intially the value of a counting semaphore is 10 Now 12 P operation are performed.
Now counting semaphore value = -2
"x" V operations were performed on this semaphore and final value of counting semaphore = 7
i.e x + (-2) = 7
x = 9.

CPU Scheduling

105.Consider three processes (process id 0, 1, 2 respectively) with compute time bursts 2, 4 and 8 time units. All processes arrive at time zero. Consider the longest remaining time first (LRTF) scheduling algorithm. In LRTF ties are broken by giving priority to the process with the lowest process id. The average turn around time is: [A]
(A) 13 units **(B)** 14 units **(C)** 15 units **(D)** 16 units

Explanation: Let the processes be p0, p1 and p2. These processes will be executed in following order.

p2 p1 p2 p1 p2 p0 p1 p2 p0 p1 p2

0 4 5 6 7 8 9 10 11 12 13 14

Turn around time of a process is total time between submission of the process and its completion.
Turn around time of p0 = 12 (12-0)

Turn around time of p1 = 13 (13-0)
Turn around time of p2 = 14 (14-0)

Average turn around time is (12+13+14)/3 = 13.

106.Consider three processes, all arriving at time zero, with total execution time of 10, 20 and 30 units, respectively. Each process spends the first 20% of execution time doing I/O, the next 70% of time doing computation, and the last 10% of time doing I/O again. The operating system uses a shortest remaining compute time first scheduling algorithm and schedules a new process either when the running process gets blocked on I/O or when the running process finishes its compute burst. Assume that all I/O operations can be overlapped as much as possible. For what percentage of time does the CPU remain idle? [B]

(A) 0% **(B)** 10.6% **(C)** 30.0% **(D)** 89.4%

Explanation: Let three processes be p0, p1 and p2. Their execution time is 10, 20 and 30 respectively. p0 spends first 2 time units in I/O, 7 units of CPU time and finally 1 unit in I/O. p1 spends first 4 units in I/O, 14 units of CPU time and finally 2 units in I/O. p2 spends first 6 units in I/O, 21 units of CPU time and finally 3 units in I/O.

idle p0 p1 p2 idle

0 2 9 23 44 47

Total time spent = 47
Idle time = 2 + 3 = 5
Percentage of idle time = (5/47)*100 = 10.6 %

107.Consider three CPU-intensive processes, which require 10, 20 and 30 time units and arrive at times 0, 2 and 6, respectively. How many context switches are needed if the operating system implements a shortest remaining time first scheduling algorithm? Do not count the context switches at time zero and at the end.[B]

(A) 1 **(B)** 2 **(C)** 3 **(D)** 4

Explanation: Let three process be P0, P1 and P2 with arrival times 0, 2 and 6 respectively and CPU burst times 10, 20 and 30 respectively. At time 0, P0 is the only available process so it runs. At time 2, P1 arrives, but P0 has the shortest remaining time, so it continues. At time 6, P2 arrives, but P0 has the shortest remaining time, so it continues. At time 10, P1 is scheduled as it is the shortest remaining time process. At time 30, P2 is scheduled. Only two context switches are needed. P0 to P1 and P1 to P2.

108.Which of the following process scheduling algorithm may lead to starvation [C]

(A) FIFO **(B)** Round Robin **(C)** Shortest Job Next **(D)** None of the above

Explanation: Shortest job next may lead to process starvation for processes which will require a long time to complete if short processes are continually added.

109.f the quantum time of round robin algorithm is very large, then it is equivalent to: [A]

(A) First in first out **(B)** Shortest Job Next
(C) Lottery scheduling **(D)** None of the above

Explanation: If time quantum is very large, then scheduling happens according to FCFS.

110.A scheduling algorithm assigns priority proportional to the waiting time of a process. Every process starts with priority zero (the lowest priority). The scheduler re-evaluates the process priorities every T time units and decides the next process to schedule. Which one of the following is TRUE if the processes have no I/O operations and all arrive at time zero? [B]

(A) This algorithm is equivalent to the first-come-first-serve algorithm
(B) This algorithm is equivalent to the round-robin algorithm.
(C) This algorithm is equivalent to the shortest-job-first algorithm..
(D) This algorithm is equivalent to the shortest-remaining-time-first algorithm

Explanation: The scheduling algorithm works as round robin with quantum time equals to T. After a process's turn comes and it has executed for T units, its waiting time becomes least and its turn comes again after every other process has got the token for T units.Consider the 3 processes, P1, P2 and P3 shown in the table.

Process	Arrival time	Time Units Required
P1	0	5
P2	1	7
P3	3	4

111.The completion order of the 3 processes under the policies FCFS and RR2 (round robin scheduling with CPU quantum of 2 time units) are [C]

(A)FCFS: P1, P2, P3
RR2: P1, P2, P3

(B) FCFS: P1, P3, P2
RR2: P1, P3, P2

(C)FCFS: P1, P2, P3
RR2: P1, P3, P2

(D)FCFS: P1, P3, P2

RR2: P1, P2, P3

Explanation:
FCFS is clear. In RR, time slot is of 2 units. Processes are assigned in following order

p1, p2, p1, p3, p2, p1, p3, p2, p2

112.This question involves the concept of ready queue. At t=2, p2 starts and p1 is sent to the ready queue and at t=3 p3 arrives so then the job p3 is queued in ready queue after p1. So at t=4, again p1 is executed then p3 is executed for first time at t=6.Consider the following table of arrival time and burst time for three processes P0, P1 and P2. [A]

Process	Arrival time	Burst Time
P0	0 ms	9 ms
P1	1 ms	4 ms
P2	2 ms	9 ms

The pre-emptive shortest job first scheduling algorithm is used. Scheduling is carried out only at arrival or completion of processes. What is the average waiting time for the three processes? [A]

(A) 5.0 ms **(B)** 4.33 ms **(C)** 6.33 **(D)** 7.33

113.Process P0 is allocated processor at 0 ms as there is no other process in ready queue. P0 is preempted after 1 ms as P1 arrives at 1 ms and burst time for P1 is less than remaining time of P0. P1 runs for 4ms. P2 arrived at 2 ms but P1 continued as burst time of P2 is longer than P1. After P1 completes, P0 is scheduled again as the remaining time for P0 is less than the burst time of P2.
P0 waits for 4 ms, P1 waits for 0 ms amd P2 waits for 11 ms. So average waiting time is (0+4+11)/3 = 5.
Which of the following statements are true? [D]

I. Shortest remaining time first scheduling may cause starvation

II. Preemptive scheduling may cause starvation

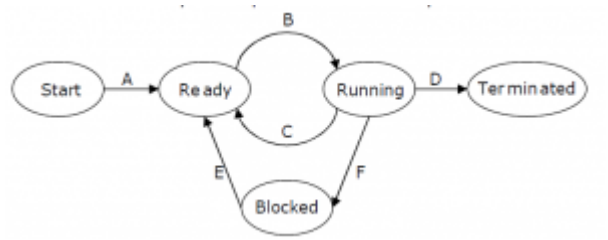
III. Round robin is better than FCFS in terms of response time

(A) I only **(B)** I and III only **(C)** II and III only **(D)** I, II and III

Explanation: I) Shortest remaining time first scheduling is a pre-emptive version of shortest job scheduling. In SRTF, job with the shortest CPU burst will be scheduled first. Because of this process, It may cause starvation as shorter processes may keep coming and a long CPU burst process never gets CPU.II) Pre-emptive just

means a process before completing its execution is stopped and other process can start execution. The stopped process can later come back and continue from where it was stopped. In pre-emptive scheduling, suppose process P1 is executing in CPU and after some time process P2 with high priority then P1 will arrive in ready queue then p1 is pre-empted and p2 will brought into CPU for execution. In this way if process which is arriving in ready queue is of higher priority then p1, then p1 is always pre-empted and it may possible that it suffer from starvation.III) round robin will give better response time then FCFS ,in FCFS when process is executing ,it executed up to its complete burst time, but in round robin it will execute up to time quantum. So Round Robin Scheduling improves response time as all processes get CPU after a specified time.

114.In the following process state transition diagram for a uniprocessor system, assume that there are always some processes in the ready state: Now consider the following statements:



I. If a process makes a transition D, it would result in another process making transition A immediately.

II. A process P2 in blocked state can make transition E while another process P1 is in running state.

III. The OS uses preemptive scheduling.

IV. The OS uses non-preemptive scheduling.

Which of the above statements are TRUE? [C]

(A) I and II **(B)** I and III **(C)** II and III **(D)** II and IV

Explanation: I is false. If a process makes a transition D, it would result in another process making transition B, not A.II is true. A process can move to ready state when I/O completes irrespective of other process being in running state or not.III is true because there is a transition from running to ready state.IV is false as the OS uses preemptive scheduling.

115.An operating system uses Shortest Remaining Time first (SRT) process scheduling algorithm. Consider the arrival times and execution times for the following processes:

Process	Execution time	Arrival time
P1	20	0
P2	25	15
P3	10	30
P4	15	45

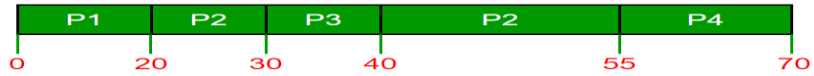
What is the total waiting time for process P2? [B]

(A) 5 **(B)** 15 **(C)** 40 **(D)** 55

Explanation:

Shortest remaining time, also known as shortest remaining time first (SRTF), is a scheduling method that is a pre-emptive version of shortest job next scheduling. In this scheduling algorithm, the process with the smallest amount of time remaining until completion is selected to execute. Since the currently executing process is the one with the shortest amount of time remaining by definition, and since that time should only reduce as execution progresses, processes will always run until they complete or a new process is added that requires a smaller amount of time.

The Gantt chart of execution of processes:



At time 0, P1 is the only process, P1 runs for 15 time units.At time 15, P2 arrives, but P1 has the shortest remaining time. So P1 continues for 5 more time units.At time 20, P2 is the only process. So it runs for 10 time units.at time 30, P3 is the shortest remaining time process. So it runs for 10 time units.at time 40, P2 runs as it is the only process. P2 runs for 5 time units.
At time 45, P3 arrives, but P2 has the shortest remaining time. So P2 continues for 10 more time units. P2 completes its execution at time 55.

Scheduling table:

PID	AT	BT	CT	TAT=CT-AT	WT=TAT-BT
P1	0	20	20	20	0
P2	15	25	55	40	15
P3	30	10	40	10	0
P4	45	15	70	25	10

AT= Arrival Time, BT=Burst Time, CT= Completion Time, TAT= Turn Around Time, WT= Waiting Time

116.As we know, turn around time is total time between submission of the process and its completion. Waiting time is the time The amount of time that is taken by a process in ready queue and waiting time is the difference between Turn around time and burst time.
Total turnaround time for P2 = Completion time – Arrival time
= 55 – 15 = 40
Total Waiting Time for P2= turn around time – Burst time
= 40 – 25 = 15
Consider three CPU-intensive processes, which require 10, 20 and 30 time units and arrive at times 0, 2 and 6, respectively. How many context switches are needed if the operating system implements a shortest remaining time first scheduling algorithm? Do not count the context switches at time zero and at the end. [B]
(A) 1 **(B)** 2 **(C)** 3 **(D)** 4
Explanation:
Shortest remaining time, also known as shortest remaining time first (SRTF), is a scheduling method that is a pre-emptive version of shortest job next scheduling. In this scheduling algorithm, the process with the smallest amount of time remaining until completion is selected to execute. Since the currently executing process is the one with the shortest amount of time remaining by definition, and since that time should only reduce as execution progresses, processes will always run until they complete or a new process is added that requires a smaller amount of time.
Solution:
Let three process be P0, P1 and P2 with arrival times 0, 2 and 6 respectively and CPU burst times 10, 20 and 30 respectively. At time 0, P0 is the only available process so it runs. At time 2, P1 arrives, but P0 has the shortest remaining time, so it continues. At time 6, P2 also arrives, but P0 still has the shortest remaining time, so it continues. At time 10, P1 is scheduled as it is the shortest remaining time process. At time 30, P2 is scheduled. Only two context switches are needed. P0 to P1 and P1 to P2.

117.Three processes A, B and C each execute a loop of 100 iterations. In each iteration of the loop, a process performs a single computation that requires tc CPU milliseconds and then initiates a single I/O operation that lasts for tio milliseconds. It is assumed that the computer where the processes execute has sufficient number of I/O devices and the OS of the computer assigns different I/O devices to each process. Also, the scheduling overhead of the OS is negligible. The processes have the following characteristics:

Process id	tc	tio
A	100 ms	500 ms
B	350 ms	500 ms
C	200 ms	500 ms

The processes A, B, and C are started at times 0, 5 and 10 milliseconds respectively, in a pure time sharing system (round robin scheduling) that uses a time slice of 50 milliseconds. The time in milliseconds at which

process C would complete its first I/O operation is ___ [B]

(A) 500 **(B)** 1000 **(C)** 2000 **(D)** 10000

Explanation:

There are three processes A, B and C that run in round robin manner with time slice of 50 ms.

Processes atart at 0, 5 and 10 miliseconds.The processes are executed in below order

A, B, C, A

50 + 50 + 50 + 50 (200 ms passed)

Now A has completed 100 ms of computations and goes for I/O now

B, C, B, C, B, C

50 + 50 + 50 + 50 + 50 + 50 (300 ms passed)

C goes for i/o at 500ms and it needs 500ms tofinish the IO.So C would complete its first IO at 1000 ms

118.An operating system uses shortest remaining time first scheduling algorithm for pre-emptive scheduling of processes. Consider the following set of processes with their arrival times and CPU burst times (in milliseconds):

Process	Arrival Time	Burst Time
P1	0	12
P2	2	4
P3	3	6
P4	8	5

The average waiting time (in milliseconds) of the processes is _____. [C]

(A) 4.5 **(B)** 5.0 **(C)** 5.5 **(D)** 6.5

Explanation:

Process	Arrival Time	Burst Time
P1	0	12
P2	2	4
P3	3	6
P4	8	5

Burst Time – The total time needed by a process from the CPU for its complete execution.

Waiting Time – How much time processes spend in the ready queue waiting their turn to get on the CPUNow, The Gantt chart for the above processes is :

P1 - 0 to 2 milliseconds
P2 - 2 to 6 milliseconds

P3 - 6 to 12 milliseconds

P4 - 12 to 17 milliseconds

P1 - 17 to 27 milliseconds

Process p1 arrived at time 0, hence cpu started executing it.After 2 units of time P2 arrives and burst time of P2 was 4 units, and the remaining time of the process p1 was 10 units,hence cpu started executing P2, putting P1 in waiting state(Pre-emptive and Shortest remaining time first scheduling).Due to P1's highest remaining time it was executed by the cpu in the end.

Now calculating the waiting time of each process:

P1 -> 17 -2 = 15

P2 -> 0

P3 -> 6 - 3 = 3

P4 -> 12 - 8 = 4

Hence total waiting time of all the processes is

= 15+0+3+4=22

Total no of processes = 4

Average waiting time = 22 / 4 = 5.5

119.Consider the following set of processes, with the arrival times and the CPU-burst times given in milliseconds

Process	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	3
P4	4	1

What is the average turnaround time for these processes with the preemptive shortest remaining processing time first (SRPT) algorithm ? [A]

(A) 5.50 **(B)** 5.75 **(C)** 6.00 **(D)** 6.25

Explanation: The following is Gantt Chart of execution

P1	P2	P4	P3	P1
1	4	5	8	12

Turn Around Time = Completion Time – Arrival Time

Avg Turn Around Time = (12 + 3 + 6+ 1)/4 = 5.50

120.A uni-processor computer system only has two processes, both of which alternate 10ms CPU bursts with 90ms I/O bursts. Both the processes were created at nearly the same time. The I/O of both processes can proceed in parallel. Which of the following scheduling strategies will result in the least CPU utilization (over a long period of time) for this system ? [D]

(A) First come first served scheduling **(B)** Shortest remaining time first scheduling

(C) Static priority scheduling with different priorities for the two processes

(D) Round robin scheduling with a time quantum of 5 ms

Explanation: When Round Robin scheduling is used

We are given that the time slice is 5ms. Consider process P and Q.Say P utilizes 5ms of CPU and then Q utilizes 5ms of CPU. Hence after 15ms P starts with I/O And after 20ms Q also starts with I/O. Since I/O can be done in parallel, P finishes I\O at 105th ms (15 + 90) and Q finishes its I\O at 110th ms (20 + 90). Therefore we can see that CPU remains idle from 20th to 105th ms.That is when Round Robin scheduling is used,Idle time of CPU =

85ms,CPU Utilization = 20/105 = 19.05%.When First Come First Served scheduling scheduling or Shortest Remaining Time First is used Say P utilizes 10ms of CPU and then starts its I/O. At 11th ms Q starts processing. Q utilizes 10ms of CPU.
P completes its I/O at 100ms (10 + 90)
Q completes its I/O at 110ms (20 + 90)
At 101th ms P again utilizes CPU. Hence,Idle time of CPU = 80ms,CPU Utilization = 20/100 = 20%
121.Since only two processes are involved and I\O time is much more than CPU time, "Static priority scheduling with different priorities" for the two processes reduces to FCFS or Shortest remaining time first.Therefore, Round robin will result in least CPU utilization.Which of the following scheduling algorithms is non-preemptive? [B]

(A) Round Robin **(B)** First-In First-Out
(C) Multilevel Queue Scheduling **(D)** Multilevel Queue Scheduling with Feedback

Explanation: Round Robin – Preemption takes place when the time quantum expiresFirst in First Out – No Preemption, the process once started completes before the other process takes over.Multi Level Queue Scheduling – Preemption takes place when a process of higher priority arrivesMulti Level Queue Scheduling with Feedback – Preemption takes a place when process of higher priority arrives or when the quantum of high priority queue expires and we need to move the process to low priority queue.So, B is the correct choice.

122.Consider a set of n tasks with known runtimes r1, r2, …. rn to be run on a uniprocessor machine. Which of the following processor scheduling algorithms will result in the maximum throughput? [B]

(A) Round-Robin **(B)** Shortest-Job-First
(C) Highest-Response-Ratio-Next **(D)** First-Come-First-Served

Explanation:
Throughput means total number of tasks executed per unit time i.e. sum of waiting time and burst time.Shortest job first scheduling is a scheduling policy that selects the waiting process with the smallest execution time to execute next.Thus, in shortest job first scheduling, shortest jobs are executed first. This means CPU utilization is maximum. So, maximum number of tasks are completed.
123.Consider a uniprocessor system executing three tasks T1, T2 and T3, each of which is composed of an infinite sequence of jobs (or instances) which arrive periodically at intervals of 3, 7 and 20 milliseconds, respectively. The priority of each task is the inverse of its period and the available tasks are scheduled in order of priority, with the highest priority task scheduled first. Each instance of T1, T2 and T3 requires an execution time of 1, 2 and 4 milliseconds, respectively. Given that all tasks initially arrive at the beginning of the 1st milliseconds and task preemptions are allowed, the first instance of T3 completes its execution at the end of _____ milliseconds. [C]

(A) 5 **(B)** 10 **(C)** 12 **(D)** 15

Explanation: Periods of T1, T2 and T3 are 3ms, 7ms and 20ms.Since priority is inverse of period, T1 is the highest priority task, then T2 and finally T3.Every instance of T1 requires 1ms, that of T2 requires 2ms and that of T3 requires 4msInitially all T1, T2 and T3 are ready to get processor, T1 is preferredSecond instances of T1, T2, and T3 shall arrive at 3, 7, and 20 respectively.Third instance of T1, T2 and T3 shall arrive at 6, 14, and 40 respectively.

Time-Interval Tasks

0-1	T1
1-2	T2
2-3	T2
3-4	T1 [Second Instance of T1 arrives]
4-5	T3
5-6	T3

6-7 T1 [Third Instance of T1 arrives]

[Therefore T3 is preempted]

7-8 T2 [Second instance of T2 arrives]

8-9 T2

9-10 T1 [Fourth Instance of T1 arrives]

10-11 T3

11-12 T3 [**First Instance of T3 completed**]

124.The maximum number of processes that can be in Ready state for a computer system with n CPUs is[D]

(A) n **(B)** n^2 **(C)** 2n **(D)** Independent of n

Explanation: The size of ready queue doesn't depend on number of processes. A single processor system may have a large number of processes waiting in ready queue.

125.For the processes listed in the following table, which of the following scheduling schemes will give the lowest average turnaround time? [C]

Process	Arrival Time	Processing Time
A	0	3
B	1	6
C	4	4
D	6	2

(A) First Come First Serve **(B)** Non-preemptive Shortest Job First

(C) Shortest Remaining Time **(D)** Round Robin with Quantum value two

Explanation: Turnaround time is the total time taken between the submission of a program/process/thread/task (Linux) for execution and the return of the complete output to the customer/user.Turnaround Time = Completion Time – Arrival Time.

FCFS = First Come First Serve (A, B, C, D)

SJF = Non-preemptive Shortest Job First (A, B, D, C)

SRT = Shortest Remaining Time (A(3), B(1), C(4), D(2), B(5))

RR = Round Robin with Quantum value 2 (A(2), B(2), A(1),C(2),B(2),D(2),C(2),B(2)

Pr	Arr.Time	P.Time	FCFS	SJF	SRT	RR
A	0	3	3-0=3	3-0=3	3-0=3	5-0=5
B	1	6	9-1=8	9-1=8	15-1=14	15-1=14
C	4	4	13-4=9	15-4=11	8-4=4	13-4=9
D	6	2	15-6=9	11-6=5	10-6=4	11-6=5
Average			7.25	6.75	6.25	8.25

126.Shortest Remaining Time produces minimum average turn-around time.

Which of the following is FALSE about SJF (Shortest Job First Scheduling)? [D]

S1: It causes minimum average waiting time

S2: It can cause starvation

(A) Only S1 **(B)** Only S2 **(C)** Both S1 and S2 **(D)** Neither S1 nor S2

Explanation:

1. Both SJF and Shortest Remaining time first algorithms may cause starvation. Consider a situation when long process is there in ready queue and shorter processes keep coming.
2. SJF is optimal in terms of average waiting time for a given set of processes, but problems with SJF is how to know/predict time of next job.

127.Two concurrent processes P1 and P2 use four shared resources R1, R2, R3 and R4, as shown below.

P1	P2
Compute:	Compute;
Use R1;	Use R1;
Use R2;	Use R2;
Use R3;	Use R3;.
Use R4;	Use R4;

Both processes are started at the same time, and each resource can be accessed by only one process at a time The following scheduling constraints exist between the access of resources by the processes:

- P2 must complete use of R1 before P1 gets access to R1
- P1 must complete use of R2 before P2 gets access to R2.
- P2 must complete use of R3 before P1 gets access to R3.
- P1 must complete use of R4 before P2 gets access to R4.

128.There are no other scheduling constraints between the processes. If only binary semaphores are used to enforce the above scheduling constraints, what is the minimum number of binary semaphores needed [B]

(A) 1 **(B)** 2 **(C)** 3 **(D)** 4

Explanation:

We use two semaphores : A and B.A is initialized to 0 and B is initialized to 1.

P1:

```
Compute;
Wait(A);
Use R1;
Use R2;
Signal(B);
Wait(A);
Use R3;
Use R4;
Signal(B);
```

P2:

```
Compute;
Wait(B);
Use r1;
Signal(A);
Wait(B);
Use R2;
Use R3;
Signal(A);
Wait(B);
Use R4;
Signal(B);
```

In process p1, initially control will be stuck in while loop of Wait(A) because A = 0.

In process p2, Wait(B) decrements the value of B to 0 . Now, P2 uses the resource R1 and increments the

value to A to 1 so that process P1 can enter its critical section and use resource R1.Thus, P2 will complete use of R1 before P1 gets access to R1.Now, in P2 values of B = 0. So, P2 can not use resource R2 till P1 uses R2 and calls function Signal(B) to increment the value of B to 1.Thus, P1 will complete use of R2 before P2 gets access to R2.Now, semaphore A = 0. So, P1 can not execute further and gets stuck in while loop of function Wait(A).Process P2 uses R3 and increments the value of semaphore A to 1.Now, P1 can enter its critical section to use R3.Thus, P2 will complete use of R3 before P1 gets access to R3.Now, P1 will use R4 and increments the value of B to 1 so that P2 can enter is critical section to use R4.Thus, P1 will complete use of R4 before P2 gets access to R4.Thus, option (B) is correct.

129.We wish to schedule three processes P1, P2 and P3 on a uniprocessor system. The priorities, CPU time requirements and arrival times of the processes are as shown below.

Process	Priority	CPU time required	Arrival time (hh:mm:ss)
P1	10(highest)	20 sec	00:00:05
P2	9	10 sec	00:00:03
P3	8 (lowest)	15 sec	00:00:00

We have a choice of preemptive or non-preemptive scheduling. In preemptive scheduling, a late-arriving higher priority process can preempt a currently running process with lower priority. In non-preemptive scheduling, a late-arriving higher priority process must wait for the currently executing process to complete before it can be scheduled on the processor.
What are the turnaround times (time from arrival till completion) of P2 using preemptive and non-preemptive scheduling respectively. [D]
(A) 30 sec, 30 sec **(B)** 30 sec, 10 sec **(C)** 42 sec, 42 sec **(D)** 30 sec, 42 sec
130.Consider an arbitrary set of CPU-bound processes with unequal CPU burst lengths submitted at the same time to a computer system. Which one of the following process scheduling algorithms would minimize the average waiting time in the ready queue?[A]

(A) Shortest remaining time first

(B) Round-robin with time quantum less than the shortest CPU burst

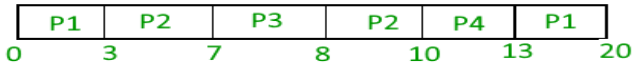
(C) Uniform random (D) Highest priority first with priority proportional to CPU burst length

Explanation: Turnaround time is the total time taken by the process between starting and the completion and waiting time is the time for which process is ready to run but not executed by CPU scheduler. As we know, in all CPU Scheduling algorithms, shortest job first is optimal i.ie. it gives minimum turn round time, minimum average waiting time and high throughput and the most important thing is that shortest remaining time first is the pre-emptive version of shortest job first. shortest remaining time first scheduling algorithm may lead to starvation because If the short processes are added to the cpu scheduler continuously then the currently running process will never be able to execute as they will get pre-empted but here all the processes are arrived at same time so there will be no issue such as starvation.So, the answer is Shortest remaining time first, which is answer (A).

131.Consider the following processes, with the arrival time and the length of the CPU burst given in milliseconds. The scheduling algorithm used is preemptive shortest remaining-time first.

Process	Arrival Time	Burst Time
P_1	0	10
P_2	3	6
P_3	7	1
P_4	8	3

The average turn around time of these processes is _____ milliseconds. [A]
Note : This question was asked as Numerical Answer Type.
(A) 8.25 **(B)** 10.25 **(C)** 6.35 **(D)** 4.25
Explanation: PreEmptive Shortest Remaining time first scheduling, i.e. that processes will be scheduled on the CPU which will be having least remaining burst time(required time at the CPU).
132.The processes are scheduled and executed as given in the below Gantt chart.



Gantt chart for the process getting scheduled and executed at the CPU

Turn Around Time(TAT) = Completion Time(CT) – Arrival Time(AT)

TAT for P1 = 20 – 0 = 20
TAT for P2 = 10 – 3 = 7

TAT for P3 = 8- 7 = 1

TAT for P4 = 13 – 8 = 5

Hence, Average TAT = Total TAT of all the processes / no of processes = (20 + 7 + 1 + 5) / 4 = 33 / 4 = 8.25

Thus, A is the correct choice.

133.Consider n jobs J_1, J_2,……J_n such that job J_i has execution time t_i and a non-negative integer weight w_i. The

weighted mean completion time of the jobs is defined to be , where T_i is the completion time of job J_i. Assuming that there is only one processor available, in what order must the jobs be executed in order to minimize the weighted mean completion time of the jobs? [D]
(A) Non-decreasing order of t_i **(B)** Non-increasing order of w_i
(C) Non-increasing order of $w_i t_i$ **(D)** None-increasing order of w_i/t_i
134.Assume every process requires 3 seconds of service time in a system with single processor. If new processes are arriving at the rate of 10 processes per minute, then estimate the fraction of time CPU is busy in system? [C]
(A) 20% **(B)** 30% **(C)** 50% **(D)** 60%
Explanation:10 processes -> 1 min
1 process-> 1/10 min = 6 sec (Arrival rate).Each process -> 3 sec service time 3/6 * 100 = 50% of time CPU is busy.

135.The most optimal scheduling algorithm is : [D]
(A) First come first serve (FCFS) **(B)** Shortest Job First (SJF)
(C) Round Robin (RR) **(D)** None of the above

Explanation: For each specific problem, the efficient algorithms could be different ones. Hence, the most optimal algorithm can't be determined.

136.Which of the following disk strategies is likely to give the best throughput? [B]

(A) Farthest cylinder next **(B)** Nearest cylinder next

(C) First come first served **(D)** Elevator algorithm

Explanation: Nearest cylinder next is also known as shortest seek time first and in this algorithm many pages can be accessed in less amount of time.So throughput will be high so this is optimal algorithm.Option (B) is correct.

137.Consider *n* processes sharing the CPU in a round-robin fashion. Assuming that each process switch takes *s* seconds, what must be the quantum size *q* such that the overhead resulting from process switching is minimized but, at the same time, each process is guaranteed to get its turn at the CPU at least every *t* seconds ? [A]

A. $q \leq \frac{t-ns}{n-1}$

B. $q \geq \frac{t-ns}{n-1}$

C. $q \leq \frac{t-ns}{n+1}$

D. $q \geq \frac{t-ns}{n+1}$

(A) a **(B)** b **(C)** c **(D)** d

Explanation: Each process will get CPU for q seconds and each process wants CPU again after t seconds.Thus, there will be (n-1) processes once after current process gets CPU again. Each process takes s seconds for context switch.(Qp1)(s)(Qp2)(s)(Qp3)(s)(Qp1)

It can be seen that since P1 left and arrived again, there have been n context switches and (n-1) processes. Thus, equation will be:

q*(n-1) + n*s <= t

q*(n-1) <= t - n*s

q <= (t-n.s) / (n-1)

138.The correct matching for the following pairs is

(A) Disk Scheduling (1) Round robin

(B) Batch Processing (2) SCAN

(C) Time sharing (3) LIFO

(D) Interrupt processing (4) FIFO

Codes:

	A	B	C	D
a	3	4	2	1
b	4	3	2	1
c	2	4	1	3
d	3	4	3	2

(A) a **(B)** b **(C)** c **(D)** d

Explanation: Round-Robin is also called Time-sharing.

Disk Scheduling Algorithms are used to reduce the total seek time of any request. SCAN is one of the Algorithms.

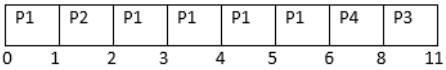
139.Interrupt processing is LIFO because when we are processing an interrupt, we disable the interrupts originating from lower priority devices so lower priority interrupts cannot be raised. If an interrupt is detected then it means that it has higher priority than currently executing interrupt so this new interrupt will preempt

the current interrupt so, LIFO.When an interrupt occurs, an operating system [D]
(A) ignores the interrupt
(B) always changes state of interrupted process to 'blocked' and schedules another process
(C) always resumes execution of interrupted process after processing the interrupt
(D) may change the state of interrupted process to 'blocked'and schedule another process
Explanation: Scheduler decides that the interrupted process will complete execution or some other process will be executed.If the interrupt signaled an I/O completion event, and at the same time a high priority process came into Ready state then the scheduler block the interrupted process and dispatch the high priority process in the running state. If low priority process comes into Ready state then scheduler dispatch the interrupted process.

140.Four jobs to be executed on a single processor system arrive at time 0 in the order A, B, C, D. Their burst CPU time requirements are 4, 1, 8, 1 time units respectively. The completion time of A under round robin scheduling with time slice of one time unit is [D]
(A) 10 **(B)** 4 **(C)** 8 **(D)** 9
141.Consider the following CPU processes with arrival times (in milliseconds) and length of CPU bursts (in milliseconds) except for process P4 as given below:

Process	Arrival Time	Burst Time
P1	0	5
P2	1	1
P3	3	3
P4	4	x

If the average waiting time across all processes is 2 milliseconds and pre-emptive shortest remaining time first scheduling algorithm is used to schedule the processes, then find the value of x ? [B]
(A) 1 **(B)** 2 **(C)** 4 **(D)** None of these
Explanation: If we take value of x is 2, then we have gantt chart as



So, completion time of P1, P2, P3, and P4 are 6, 2, 11, and 8 respectively.Turn around time of P1, P2, P3, and P4 are 6, 1, 8, and 4 respectively.Waiting time of P1, P2, P3, and P4 are 1, 0, 5, and 2 resectively.Therefore, average waiting time = (1+0+5+2) / 4 = 8/2 = 2

142.Which module gives control of the CPU to the process selected by the short – term schedular ? [A]

(A) Dispatcher **(B)** Interrupt **(C)** Schedular **(D)** Threading
Explanation: There are three type of scheduler:
1. Short-term scheduler 2.Mid-term scheduler 3.Long-term scheduler
Dispatcher is responsible for handovering the control of CPU to the newly selected process by the Short-term scheduler.
143Consider the following scheduling

A. Gang scheduling	s. Guaranteed scheduling
B. Rate Monotonic scheduling	t. Thread scheduling
C. Fair share	u. Real time

scheduling scheduling

Matching the table in the order A, B, C gives [A]

(A) t, u, s **(B)** s, t, u **(C)** u, s, t **(D)** u, t, s

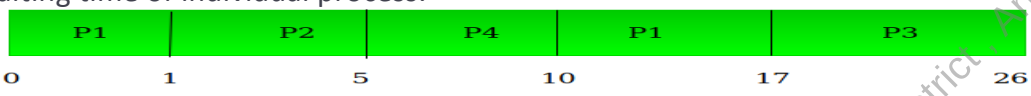
144.Consider the following four processes with the arrival time and length of CPU burst given in milliseconds :

Process	Arrival Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

The average waiting time for preemptive SJF scheduling algorithm is _____. [A]

(A) 6.5 **(B)** 7.5 **(C)** 6.75 **(D)** 7.75

Explanation: First we will make gantt chart of given process then we will calculate turn around time and waiting time of individual process.



Process id	CT	TAT	WT
P1	17	17	9
P2	5	4	0
P3	26	24	15
P4	10	7	2

Now we have to calculate average waiting time for schedule:

avg waiting time = wt(P1 + P2 + P3 + P4)/number of process.

ie. (9 + 0 + 15 + 2) / 4

= 26 / 4

= 6.5

145.Which of the following scheduling algorithms may cause starvation ? [B]

a. First-come-first-served b. Round Robin c. Priority d. Shortest process next

e. Shortest remaining time first

(A) a, c and e **(B)** c, d and e **(C)** b, d and e **(D)** b, c and d

Explanation:

1. In First Come First Serve(FCFS) if a process with a very large **Burst Time** comes before other processes, the other process will have to wait for a long time but it is clear that other process will definitely get their chance to execute, so it will not suffer from starvation.
2. In Round Robin there is a fixed time quant and every process will get their chance to be executed, so no starvation is here.
3. In Priority based scheduling if higher priority process keep on coming then low priority process will suffer from starvation.
4. In Shortest Job First(SJF) if process with short process time keep on coming continuously then process with higher burst time will do wait and suffer from starvation.
5. In Shortest remaining time first(SRTF) process with shortest burst time will execute first because of this process with high burst time may suffer from starvation.

146.Five jobs A, B, C, D and E are waiting in Ready Queue. Their expected runtimes are 9, 6, 3, 5 and x respectively. All jobs entered in Ready queue at time zero. They must run in _____ order to minimize average response time if 3 < x < 5. [B]

(A) B, A, D, E, C **(B)** C, E, D, B, A **(C)** E, D, C, B, A **(D)** C, B, A, E, D

Explanation: We will solve it by minimizing the avg Waiting Time and Take x = 4

BT = Burst or Execution time

CT = completion time

P = Process

AWT = Average Waiting Time

AT = Arrival time

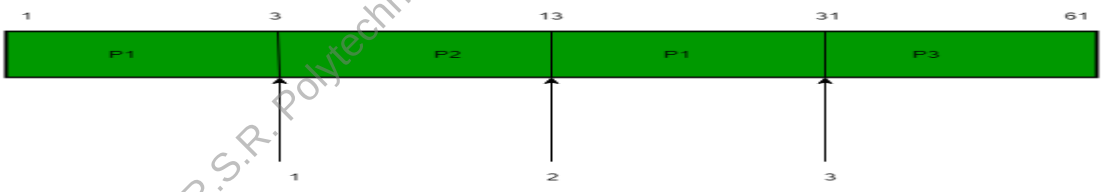
WT = Waiting time

			Option A		Option B		Option C		Option D	
P	AT	BT	CT_A	WT_A	CT_B	WT_B	CT_C	WT_C	CT_D	WT_D
A	0	9	15	6	27	18	27	18	18	9
B	0	6	6	0	18	12	18	12	9	3
C	0	3	27	24	3	0	12	9	3	0
D	0	5	19	14	12	7	9	4	27	22
E	0	4	23	19	7	3	4	0	22	18
				63/5		40/5		43/5		52/5
				AWT=12.6		AWT=8		AWT=8.6		AWT=10.4

We get minimum average waiting time from the sequence given in option (B)

147.Consider three CPU intensive processes P_1, P_2, P_3 which require 20, 10 and 30 units of time, arrive at times 1, 3 and 7 respectively. Suppose operating system is implementing Shortest Remaining Time first (preemptive scheduling) algorithm, then _____ context switches are required (suppose context switch at the beginning of Ready queue and at the end of Ready queue are not counted). [A]

(A) 3 **(B)** 2 **(C)** 4 **(D)** 5



Explanation:

There are 3 context switch.

148.Which of the following is not an optimization criterion in the design of a CPU scheduling algorithm? [A]

(A) Minimum CPU utilization **(B)** Maximum throughput

(C) Minimum turnaround time **(D)** Minimum waiting time

Explanation: Minimum CPU utilization is not an optimization criterion as various optimization techniques and scheduling algorithms are used to bring the best CPU performance.

149.A scheduling Algorithm assigns priority proportional to the waiting time of a process. Every process starts with priority zero (lowest priority). The scheduler reevaluates the process priority for every 'T' time units and decides next process to be scheduled. If the process have no I/O operations and all arrive at time zero, then the scheduler implements _____ criteria. [B]

(A) Priority scheduling **(B)** Round Robin Scheduling **(C)** Shortest Job First **(D)** FCFS

Explanation: A scheduling Algorithm assigns priority proportional to the waiting time of a process. Every process starts with priority zero (lowest priority). The scheduler reevaluates the process priority for every 'T' time units and decides next process to be scheduled. If the process have no I/O operations and all arrive at time zero, then the scheduler implements **Round Robin Scheduling** criteria.

150.A system has four processes and five allocatable resources. The current allocation and maximum needs are as follows:

	Allocated	Maximum	Available
Process A	1 0 2 1 1	1 1 2 1 3	0 0 x 1 1
Process B	2 0 1 1 0	2 2 2 1 0	
Process C	1 1 0 1 0	2 1 3 1 0	
Process D	1 1 1 1 0	1 1 2 2 1	

The smallest value of x for which the above system in safe state is _____. [D]

(A) 1 **(B)** 3 **(C)** 2 **(D)** Not safe for any value of x.

Explanation:

	Allocated	Maximum	Available	Need
Process A	1 0 2 1 1	1 1 2 1 3	0 0 x 1 1	0 1 0 0 2
Process B	2 0 1 1 0	2 2 2 1 0		0 2 1 0 0
Process C	1 1 0 1 0	2 1 3 1 0		1 0 3 0 0
Process D	1 1 1 1 0	1 1 2 2 1		0 0 1 1 1

The smallest value of x for which the above system in safe state is _____.

For x = 1 process D will execute and free 1 1 2 2 1 instances. Now none of the other process will execute.let x = 2 then process D will execute and free 1 1 3 2 1 instances. Now process C will execute and free 2 2 3 3 1 instances. With these free instances process B will execute, but process A will not execute because 5 resources needs 2 instances which will never be satisfied. That's why system is not in safe state.So, option (D) is correct.

151.In a system using single processor, a new process arrives at the rate of six processes per minute and each such process requires seven seconds of service time. What is the CPU utilization? [A]

(A) 70% **(B)** 30% **(C)** 60% **(D)** 64%

Explanation: Number of processes per minute = 6

Burst time of each process = 7 secs

CPU utilization time within a minute = 6*7 = 42 secs

% CPU utilization = useful time / total time * 100

= (42/60) * 100

= 70%

151.Which of the following statements is not true for Multi Level Feedback Queue processor scheduling algorithm? [C]

(A) Queues have different priorities **(B)** Each queue may have different scheduling algorithm

(C) Processes are permanently assigned to a queue

(D) This algorithm can be configured to match a specific system under design

Explanation: For Multi Level Feedback Queue processor scheduling algorithm:

Queues have different priorities.Each queue may have different scheduling algorithm.Processes are not permanently assigned to a queue.This algorithm can be configured to match a specific system under design

152.Consider the following justifications for commonly using the two-level CPU scheduling:

I. It is used when memory is too small to hold all the ready processes.

II. Because its performance is same as that of the FIFO.

III. Because it facilitates putting some set of processes into memory and a choice is made from that.

IV. Because it does not allow to adjust the set of in-core processes.

Which of the following is true ? [D]

(A) I, III and IV **(B)** I and II **(C)** III and IV **(D)** I and III

Explanation: The two-level CPU scheduling is used when memory is too small to hold all the ready processes because it facilitates putting some set of processes into memory and a choice is made from that.

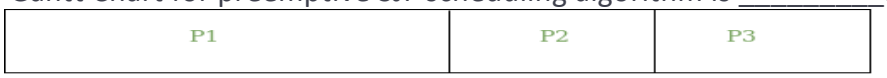
153.In a multi-user operating system, 30 requests are made to use a particular resource per hour, on an average. The probability that no requests are made in 40 minutes, when arrival pattern is a poisson distribution, is _____. [D]

(A) e^{-15} **(B)** $1 - e^{-15}$ **(C)** $1 - e^{-20}$ **(D)** e^{-20}

154.Consider the following three processes with the arrival time and CPU burst time given in milliseconds: [B]

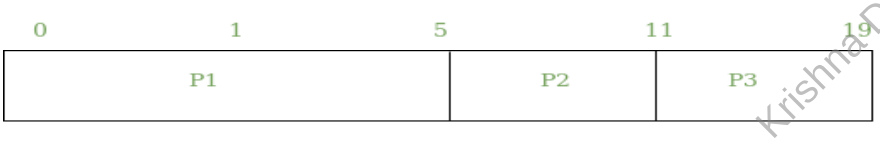
Process	Arrival Time	Burst Time
P_1	0	7
P_2	1	4
P_3	2	8

The Gantt Chart for preemptive SJF scheduling algorithm is _____.



(A)

(B)

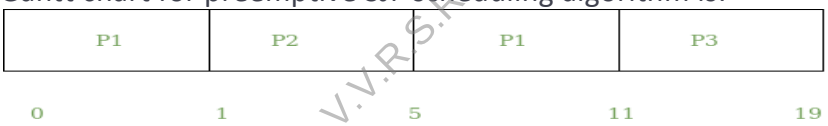


(C)

(D)

Explanation: P_1 arrive at 0 it will be served by CPU for 1 unit. After 1 unit P_2 arrived ans it is shortest So it will execute for 4 unit, after its execution There are 2 processes out of which P_1 is shortest so it will execute for 6 unit after its execution P_1 will execute for 8 unit.

Gantt chart for preemptive SJF scheduling algorithm is:



155.In which of the following scheduling criteria, context switching will never take place ? [C]

(A) ROUND ROBIN **(B)** Preemptive SJF **(C)** Non-preemptive SJF **(D)** Preemptive priority

Explanation: In Non – preemptive algorithms context switching will never take place because it doesn't allow to switch the process until it is completed fully.So, option (C) is correct.

156.Which of the following page replacement algorithms suffers from Belady's anomaly? [A]

(A) FIFO **(B)** LRU **(C)** Optimal Page Replacement **(D)** Both LRU and FIFO

Explanation: Belady's anomaly proves that it is possible to have more page faults when increasing the number of page frames while using the First in First Out (FIFO) page replacement algorithm.

157.What is the swap space in the disk used for? [B]

(A) Saving temporary html pages **(B)** Saving process data

(C) Storing the super-block **(D)** Storing device drivers

158.Increasing the RAM of a computer typically improves performance because: [C]

(A) Virtual memory increases **(B)** Larger RAMs are faster

(C) Fewer page faults occur **(D)** Fewer segmentation faults occur

Explanation: When there is more RAM, there would be more mapped virtual pages in physical memory,

hence fewer page faults. A page fault causes performance degradation as the page has to be loaded from secondary device.

159.A computer system supports 32-bit virtual addresses as well as 32-bit physical addresses. Since the virtual address space is of the same size as the physical address space, the operating system designers decide to get rid of the virtual memory entirely. Which one of the following is true? [C]

(A) Efficient implementation of multi-user support is no longer possible
(B) The processor cache organization can be made more efficient now
(C) Hardware support for memory management is no longer needed
(D) CPU scheduling can be made more efficient now

Explanation: For supporting virtual memory, special hardware support is needed from Memory Management Unit. Since operating system designers decide to get rid of the virtual memory entirely, hardware support for memory management is no longer needed

160.A CPU generates 32-bit virtual addresses. The page size is 4 KB. The processor has a translation look-aside buffer (TLB) which can hold a total of 128 page table entries and is 4-way set associative. The minimum size of the TLB tag is: [C]

(A) 11 bits **(B)** 13 bits **(C)** 15 bits **(D)** 20 bits

Explanation: Size of a page = 4KB = 2^12
Total number of bits needed to address a page frame = 32 – 12 = 20
If there are 'n' cache lines in a set, the cache placement is called n-way set associative. Since TLB is 4 way set associative and can hold total 128 (2^7) page table entries, number of sets in cache = 2^7/4 = 2^5. So 5 bits are needed to address a set, and 15 (20 – 5) bits are needed for tag.

161.Virtual memory is [C]

(A) Large secondary memory **(B)** Large main memory
(C) Illusion of large main memory **(D)** None of the above

Explanation: Virtual memory is illusion of large main memory.

162.Page fault occurs when [B]

(A) When a requested page is in memory **(B)** When a requested page is not in memory
(C) When a page is corrupted **(D)** When an exception is thrown

Explanation: Page fault occurs when a requested page is mapped in virtual address space but not present in memory.

163.Thrashing occurs when [B]

(A) When a page fault occurs **(B)** Processes on system frequently access pages not memory
(C) Processes on system are in running state **(D)** Processes on system are in waiting state

Explanation: Thrashing occurs processes on system require more memory than it has.
If processes do not have "enough" pages, the pagefault rate is very high. This leads to:
– low CPU utilization
– operating system spends most of its time swapping to disk

The above situation is called thrashing

165.What is the size of a page in KB in this computer? (GATE 2013) [C]

(A) 2 **(B)** 4 **(C)** 8 **(D)** 16

Explanation:
Let the page size is of 'x' bits.Size of T1 = 2 ^ x bytes.(This is because T1 occupies exactly one page).Now, number of entries in T1 = (2^x) / 4.(This is because each page table entry is 32 bits

or 4 bytes in size).Number of entries in T1 = Number of second level

page tables(Because each I-level page table entry stores the base address of page of II-level page table)

Total size of second level page tables = ((2^x) / 4) * (2^x)

Similarly, number of entries in II-level page tables = Number

of III level page tables = ((2^x) / 4) * ((2^x) / 4)

Total size of third level page tables = ((2^x) / 4) *

((2^x) / 4) * (2^x)

Similarly, total number of entries (pages) in all III-level

page tables = ((2^x) / 4) * ((2^x) / 4) * ((2^x) / 4)

= 2^(3x - 6)

Size of virtual memory = 2^46

Number of pages in virtual memory = (2^46) / (2^x) = 2^(46 - x)

Total number the pages in the III-level page tables = Number of pages in virtual memory

2^(3x - 6) = 2^(46 - x)

3x - 6 = 46 - x

4x = 52

x = 13

That means, page size is of 13 bits or Page size = 2^13 bytes = 8 KB

166.Consider data given in the above question. What is the minimum number of page colours needed to guarantee that no two synonyms map to different sets in the processor cache of this computer? [C]

(A) 2 **(B)** 4 **(C)** 8 **(D)** 16

Explanation:

1 MB 16-way set associative virtually indexed physically tagged cache(**VIPT**).
The cache block size is 64 bytes.No of blocks is 2^20/2^6 = 2^14.
No of sets is 2^14/2^4 = 2^10.
VA(46)
+------------------------------+
tag(30) , Set(10) , block offset(6)
+------------------------------+
In VIPT if the no. of bits of page offset =
(Set+block offset) then only one page color is sufficient.
but we need 8 colors because the number bits where the cache set index and
physical page number over lap is 3 so 2^3 page colors is required.(option c is ans).

167.Consider the virtual page reference string
1, 2, 3, 2, 4, 1, 3, 2, 4, 1
On a demand paged virtual memory system running on a computer system that main memory size of 3 pages frames which are initially empty. Let LRU, FIFO and OPTIMAL denote the number of page faults under the corresponding page replacements policy. Then [B]

(A) OPTIMAL < LRU < FIFO **(B)** OPTIMAL < FIFO < LRU **(C)** OPTIMAL = LRU **(D)** OPTIMAL = FIFO

Explanation:

First In First Out (FIFO)

This is the simplest page replacement algorithm. In this algorithm, operating system keeps track of all pages in the memory in a queue; oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

Optimal Page replacement:in this algorithm, pages are replaced which are not used for the longest duration of time in the future. Least Recently Used (LRU) In this algorithm page will be replaced which is least recently used.

Solution:

the virtual page reference string is 1, 2, 3, 2, 4, 1, 3, 2, 4, 1 size of main memory pages frames is 3.For FIFO: total no of page faults are 6 (depicted in red)

1	1	1	1	4	4	4	4	4	4
	2	2	2	2	1	1	1	1	1
		3	3	3	3	3	2	2	2

For optimal: total no of page faults are 5 (depicted in red)

1	1	1	1	1	1	1	1	1	1
	2	2	2	4	4	4	4	4	4
		3	3	3	3	3	2	2	2

For LRU: total no of page faults are 9 (depicted in red)

1	1	1	1	4	4	4	2	2	2
	2	2	2	2	2	3	3	3	1
		3	3	3	1	1	1	4	4

The Optimal will be 5, FIFO 6 and LRU 9. so, OPTIMAL < FIFO < LRU option (B) is correct answer.

168.Let the page fault service time be 10ms in a computer with average memory access time being 20ns. If one page fault is generated for every 10^6 memory accesses, what is the effective access time for the memory? [B]

(A) 21ns **(B)** 30ns **(C)** 23ns **(D)** 35ns

Explanation:

Let P be the page fault rate

Effective Memory Access Time = p * (page fault service time) +

(1 - p) * (Memory access time)

= (1/(10^6))* 10 * (10^6) ns +

(1 - 1/(10^6)) * 20 ns

= 30 ns (approx)

169.A system uses FIFO policy for page replacement. It has 4 page frames with no pages loaded to begin with. The system first accesses 100 distinct pages in some order and then accesses the same 100 pages but now in the reverse order. How many page faults will occur? [A]

(A) 196 **(B)** 192 **(C)** 197 **(D)** 195

170.in which one of the following page replacement policies, Belady's anomaly may occur? [A]

(A) FIFO **(B)** Optimal **(C)** LRU **(D)** MRU

Explanation: Belady's anomaly proves that it is possible to have more page faults when increasing the number of page frames while using the First in First Out (FIFO) page replacement algorithm.

171.The essential content(s) in each entry of a page table is / are [B]

(A) Virtual page number **(B)** Page frame number

(C) Both virtual page number and page frame number **(D)** Access right information

Explanation:

A page table entry must contain Page frame number. Virtual page number is typically used as index in page table to get the corresponding page frame number.

172.A multilevel page table is preferred in comparison to a single level page table for translating virtual address to physical address because [B]

(A) It reduces the memory access time to read or write a memory location.

(B) It helps to reduce the size of page table needed to implement the virtual address space of a process.

(C) It is required by the translation lookaside buffer.

(D) It helps to reduce the number of page faults in page replacement algorithms.

Explanation: The size of page table may become too big (See this) to fit in contiguous space. That is why page tables are typically divided in levels.

173.A processor uses 36 bit physical addresses and 32 bit virtual addresses, with a page frame size of 4 Kbytes. Each page table entry is of size 4 bytes. A three level page table is used for virtual to physical address translation, where the virtual address is used as follows

- Bits 30-31 are used to index into the first level page table
- Bits 21-29 are used to index into the second level page table
- Bits 12-20 are used to index into the third level page table, and
- Bits 0-11 are used as offset within the page

The number of bits required for addressing the next level page table (or page frame) in the page table entry of the first, second and third level page tables are respectively. [D]

(A) 20, 20 and 20 **(B)** 24, 24 and 24 **(C)** 24, 24 and 20 **(D)** 25, 25 and 24

Explanation: Virtual address size = 32 bits

Physical address size = 36 bits.Physical memory size = 2^36 bytes.Page frame size = 4K bytes = 2^12 bytes

174.A virtual memory system uses First In First Out (FIFO) page replacement policy and allocates a fixed number of frames to a process. Consider the following statements:

P: Increasing the number of page frames allocated to a process sometimes increases the page fault rate.

Q: Some programs do not exhibit locality of reference.

Which one of the following is TRUE? [B]

(A) Both P and Q are true, and Q is the reason for P **(B)** Both P and Q are true, but Q is not the reason for P.

(C) P is false, but Q is true **(D)** Both P and Q are false

Explanation: First In First Out Page Replacement Algorithms: This is the simplest page replacement algorithm. In this algorithm, operating system keeps track of all pages in the memory in a queue, oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.FIFO Page replacement algorithms suffers from Belady's anomaly :Belady's anomaly states that it is possible to have more page faults when increasing the number of page frames.Solution:**Statement P:** Increasing the number of page frames allocated to a process sometimes increases the page fault rate.Correct, as FIFO page replacement algorithm suffers from belady's anomaly which states above statement.**Statement Q:** Some programs do not exhibit locality of reference. Correct, Locality often occurs because code contains loops that tend to reference arrays or other data structures by indices. So we can write a program does not contain loop and do not exhibit locality of reference.So, both statement P and Q are correct but Q is not the reason for P as Belady's Anomaly occurs for some specific patterns of page references.

176.A process has been allocated 3 page frames. Assume that none of the pages of the process are available in the memory initially. The process makes the following sequence of page references (reference string): 1, 2, 1, 3, 7, 4, 5, 6, 3, 1.If optimal page replacement policy is used, how many page faults occur for the above reference string? [A]

(A) 7 **(B)** 8 **(C)** 9 **(D)** 10

Explanation: Optimal replacement policy looks forward in time to see which frame to replace on a page fault.

1 23 -> 1,2,3 //page faults

173 ->7

143 ->4

153 -> 5

163 -> 6

Total=7

177.Consider the data given in above question.Least Recently Used (LRU) page replacement policy is a practical approximation to optimal page replacement. For the above reference string, how many more page faults occur with LRU than with the optimal page replacement policy? [C]

(A) 0 **(B)** 1 **(C)** 2 **(D)** 3

Explanation: LRU replacement policy: The page that is least recently used is being Replaced.

Given String: 1, 2, 1, 3, 7, 4, 5, 6, 3, 1

123 // 1 ,2, 3 //page faults

173 ->7

473 ->4

453 ->5

456 ->6

356 ->3

316 ->1

Total 9

178.Assume that there are 3 page frames which are initially empty. If the page reference string is 1, 2, 3, 4, 2, 1, 5, 3, 2, 4, 6, the number of page faults using the optimal replacement policy is_____. [C]

(A) 5 **(B)** 6 **(C)** 7 **(D)** 8

Explanation: In optimal page replacement replacement policy, we replace the place which is not used for longest duration in future.

Given three page frames.Reference string is **1, 2, 3, 4, 2, 1, 5, 3, 2, 4, 6**

Initially, there are three page faults and entries are 1 2 3

Page 4 causes a page fault and replaces 3 (3 is the longest distant in future), entries become

1 2 4

Total page faults = 3+1 = 4

Pages 2 and 1 don't cause any fault.

5 causes a page fault and replaces 1, entries become

5 2 4

Total page faults = 4 + 1 = 5

3 causes a page fault and replaces 5, entries become

3 2 4

Total page faults = 5 + 1 = 6

3, 2 and 4 don't cause any page fault.

6 causes a page fault.

Total page faults = 6 + 1 = 7

179.A computer has twenty physical page frames which contain pages numbered 101 through 120. Now a program accesses the pages numbered 1, 2, …, 100 in that order, and repeats the access sequence THRICE. Which one of the following page replacement policies experiences the same number of page faults as the optimal page replacement policy for this program? [D]

(A) Least-recently-used **(B)** First-in-first-out **(C)** Last-in-first-out **(D)** Most-recently-used

Explanation: The optimal page replacement algorithm swaps out the page whose next use will occur farthest in the future.In the given question, the computer has 20 page frames and initially page frames are filled with

pages numbered from 101 to 120. Then program accesses the pages numbered 1, 2, …, 100 in that order, and repeats the access sequence THRICE.
The first 20 accesses to pages from 1 to 20 would definitely cause page fault. When 21st is accessed, there is another page fault. The page swapped out would be 20 because 20 is going to be accessed farthest in future. When 22nd is accessed, 21st is going to go out as it is going to be the farthest in future.The above optimal page replacement algorithm actually works as most recently used in this case.

Iteration 1: (1-100) Not present – all replaced 1-20 in 20 frames, 21-100 in 20th frame. hence, page faults = **100**
Iteration 2: (1-19) present **|** (20-99) NOT present **|** (100) present – the replacements are done at the 19th frame hence, page faults = 100 – 19 – 1 = **80**
Iteration 3: (1-18) present **|** (19-98) NOT present **|** (99-100) present – the replacements are done at the 18th frame hence page faults = 100 – 18 – 2 = **80**
Iteration 4: (1-17) present **|** (17-97) NOT present **|** (98-100) present – the replacements are done at the 17th frame hence page faults = 100 – 17 – 3 = **80**
Total page faults = **100 + 80 + 80 +80 = 340**
Along with generating same number of page faults M.R.U also generates replacements at the same positions as in the Optimal algorithm.(Assuming the given 101-120 pages are INVALID (not belonging to the same process) or Empty).

While the LIFO replacement does not behave like Optimal replacement algorithm as it generates 343 page faults. Because from 21st page all pages are placed in 20th frame, therefore hits per iteration reduces down to 19 from the 2nd iteration of pages. Whereby

Total Page faults = **100 + 81 + 81 + 81 = 343**
180.A system uses 3 page frames for storing process pages in main memory. It uses the Least Recently Used (LRU) page replacement policy. Assume that all the page frames are initially empty. What is the total number of page faults that will occur while processing the page reference string given below? [C]
4, 7, 6, 1, 7, 6, 1, 2, 7, 2
(A) 4 **(B)** 5 **(C)** 6 **(D)** 7
Explanation: What is a Page fault ? An interrupt that occurs when a program requests data that is not currently in real memory. The interrupt triggers the operating system to fetch the data from a virtual memory and load it into RAM.
Now, 4, 7, 6, 1, 7, 6, 1, 2, 7, 2 is the reference string, you can think of it as data requests made by a program.Now the system uses 3 page frames for storing process pages in main memory. It uses the Least Recently Used (LRU) page replacement policy.

[ ] - Initially page frames are empty.i.e. no process pages in main memory.

[4] - Now 4 is brought into 1st frame (1st page fault)

Explanation: Process page 4 was requested by the program, but it was not in the main memory(in form of page frames),which resulted in a page fault, after that process page 4 was brought in the main memory by the operating system.

[4 7] - Now 7 is brought into 2nd frame (2nd page fault) - Same explanation.

[4 7 6] - Now 6 is brought into 3rd fram 3rd page fault)

[1 7 6] - Now 1 is brought into 1st frame, as 1st frame was least recently used(4th page fault). After this 7, 6 and 1 are were already present in the frames hence no replacements in pages.

[1 2 6] - Now 2 is brought into 2nd frame, as 2n frame was least recently used(5th page fault).

[1 2 7] -Now 7 is brought into 3rd frame, as 3rd frame was least recently used(6th page fault). Hence, total number of page faults(also called pf) are 6. Therefore, C is the answer.
181.Consider a paging hardware with a TLB. Assume that the entire page table and all the pages are in the

physical memory. It takes 10 milliseconds to search the TLB and 80 milliseconds to access the physical memory. If the TLB hit ratio is 0.6, the effective memory access time (in milliseconds) is _____.
(A) 120 **(B)** 122 **(C)** 124 **(D)** 118
Explanation: TLB stands for Translation Lookaside Buffer. In Virtual memory systems, the cpu generates virtual memory addresses. But, the data is stored in actual physical memory i.e. we need to place a physical memory address on the memory bus to fetch the data from the memory circuitry. So, a special table is maintained by the operating system called the Page table. This table contains a mapping between the virtual addresses and physical addresses. So, every time a cpu generates a virtual address, the operating system page table has to be looked up to find the corresponding physical address. To speed this up, there is hardware support called the TLB. The TLB is a high speed cache of the page table i.e. contains recently accessed virtual to physical translations. TLB hit ratio- A TLB hit is the no of times a virtual-to-physical address translation was already found in the TLB, instead of going all the way to the page table which is located in slower physical memory. TLB hit ratio is nothing but the ratio of TLB hits/Total no of queries into TLB. In the case that the page is found in the TLB (TLB hit) the total time would be the time of search in the TLB plus the time to access memory, so

TLB_hit_time := TLB_search_time + memory_access_time
In the case that the page is not found in the TLB (TLB miss) the total time would be the time to search the TLB (you don't find anything, but searched nontheless) plus the time to access memory to get the page table and frame, plus the time to access memory to get the data, so.**TLB_miss_time := TLB_search_time + memory_access_time + memory_access_time**
But this is in individual cases, when you want to know an average measure of the TLB performance, you use the Effective Access Time, that is the weighted average of the previous measures

EAT := TLB_miss_time * (1- hit_ratio) + TLB_hit_time * hit_ratio.

EAT := (TLB_search_time + 2*memory_access_time) * (1- hit_ratio) + (TLB_search_time + memory_access_time)* hit_ratio.
As both page table and page are in physical memory
T(eff) = hit ratio * (TLB access time + Main memory access time) +
(1 – hit ratio) * (TLB access time + 2 * main memory time)
= 0.6*(10+80) + (1-0.6)*(10+2*80)
= 0.6 * (90) + 0.4 * (170)
= 122

182.The memory access time is 1 nanosecond for a read operation with a hit in cache, 5 nanoseconds for a read operation with a miss in cache, 2 nanoseconds for a write operation with a hit in cache and 10 nanoseconds for a write operation with a miss in cache. Execution of a sequence of instructions involves 100 instruction fetch operations, 60 memory operand read operations and 40 memory operand write operations. The cache hit-ratio is 0.9. The average memory access time (in nanoseconds) in executing the sequence of instructions is _____[B]
(A) 1.26 **(B)** 1.68 **(C)** 2.4 **(D)** 4.52
Explanation:
The question is to find the time taken for,

"100 fetch operation and 60 operand red operations and 40 memory

operand write operations"/"total number of instructions".

Total number of instructions= 100+60+40 =200

Time taken for 100 fetch operations(fetch =read)

= 100*((0.9*1)+(0.1*5)) // 1 corresponds to time taken for read

// when there is cache hit

= 140 ns //0.9 is cache hit rate

Time taken for 60 read operations = 60*((0.9*1)+(0.1*5))

= 84ns

Time taken for 40 write operations = 40*((0.9*2)+(0.1*10))

= 112 ns

// Here 2 and 10 the time taken for write when there is cache

// hit and no cahce hit respectively

So,the total time taken for 200 operations is = 140+84+112 = 336ns

Average time taken = time taken per operation = 336/200 = 1.68 ns

183.A CPU generates 32-bit virtual addresses. The page size is 4 KB. The processor has a translation look-aside buffer (TLB) which can hold a total of 128 page table entries and is 4-way set associative. The minimum size of the TLB tag is: [C]

(A) 11 bits **(B)** 13 bits **(C)** 15 bits **(D)** 20 bits

Explanation:

Virtual Memory would not be very effective if every memory address had to be translated by looking up the associated physical page in memory. The solution is to cache the recent translations in a Translation Lookaside Buffer (TLB). A TLB has a fixed number of slots that contain page table entries, which map virtual addresses to physical addresses.

Solution

Size of a page = 4KB = 2^12 means 12 offset bits CPU generates 32-bit virtual addresses

Total number of bits needed to address a page frame = 32 – 12 = 20

If there are 'n' cache lines in a set, the cache placement is called n-way set associative. Since TLB is 4 way set associative and can hold total 128 (2^7) page table entries, number of sets in cache = 2^7/4 = 2^5. So 5 bits are needed to address a set, and 15 (20 – 5) bits are needed for tag.Option (C) is the correct answer.

184.A computer system supports 32-bit virtual addresses as well as 32-bit physical addresses. Since the virtual address space is of the same size as the physical address space, the operating system designers decide to get rid of the virtual memory entirely. Which one of the following is true? [C]

(A) Efficient implementation of multi-user support is no longer possible

(B) The processor cache organization can be made more efficient now

(C) Hardware support for memory management is no longer needed

(D) CPU scheduling can be made more efficient now

185.The minimum number of page frames that must be allocated to a running process in a virtual memory environment is determined by [A]

(A) the instruction set architecture **(B)** page size

(C) physical memory size **(D)** number of processes in memory

Explanation:

There are two important tasks in virtual memory management: a page-replacement strategy and a frame-allocation strategy. Frame allocation strategy says gives the idea of minimum number of frames which should be allocated. The absolute minimum number of frames that a process must be allocated is dependent on system architecture, and corresponds to the number of pages that could be touched by a single (machine) instruction.So, it is instruction set architecture i.e. option (A) is correct answer.

186.Consider a system with a two-level paging scheme in which a regular memory access takes 150 nanoseconds, and servicing a page fault takes 8 milliseconds. An average instruction takes 100 nanoseconds of CPU time, and two memory accesses. The TLB hit ratio is 90%, and the page fault rate is one in every 10,000 instructions. What is the effective average instruction execution time? [D]

(A) 645 nanoseconds **(B)** 1050 nanoseconds **(C)** 1215 nanoseconds **(D)** 1230 nanoseconds

Explanation: Please note that page fault rate is given one page fault per 10,000 instructions. Since there are

two memory accesses per instruction, so we need double address translation time for average instruction execution time. Also, there are 2 page table accessed if TLB miss occurred. TLB access assumed as 0.Therefore,Average Instruction execution time = Average CPU execution time + Average time for getting data(instruction operands from memory for each instruction)
= Average CPU execution time + Average address translation time for each instruction + Average memory fetch time for each instruction + Average page fault time for each instruction

$= 100 + 2\times(0.9\times(0)+0.1\times(2\times150)) + 2\times150 + 1 /10000 \times 8 \times 10^6$

= 100 + 60 + 300 + 800
= 1260 ns

So, none option is correct.
187.In a system with 32 bit virtual addresses and 1 KB page size, use of one-level page tables for virtual to physical address translation is not practical because of [C]
(A) the large amount of internal fragmentation **(B)** the large amount of external fragmentation
(C) the large memory overhead in maintaining page tables
(D) the large computation overhead in the translation process
188.In a system with 32 bit virtual addresses and 1 KB page size, use of one-level page tables for virtual to physical address translation is not practical because of [C]
(A) the large amount of internal fragmentation
(B) the large amount of external fragmentation
(C) the large memory overhead in maintaining page tables
(D) the large computation overhead in the translation process
189.A processor uses 2-level page tables for virtual to physical address translation. Page tables for both levels are stored in the main memory. Virtual and physical addresses are both 32 bits wide. The memory is byte addressable. For virtual to physical address translation, the 10 most significant bits of the virtual address are used as index into the first level page table while the next 10 bits are used as index into the second level page table. The 12 least significant bits of the virtual address are used as offset within the page. Assume that the page table entries in both levels of page tables are 4 bytes wide. Further, the processor has a translation look-aside buffer (TLB), with a hit rate of 96%. The TLB caches recently used virtual page numbers and the corresponding physical page numbers. The processor also has a physically addressed cache with a hit rate of 90%. Main memory access time is 10 ns, cache access time is 1 ns, and TLB access time is also 1 ns.Assuming that no page faults occur, the average time taken to access a virtual address is approximately (to the nearest 0.5 ns) [D]
(A) 1.5 ns **(B)** 2 ns **(C)** 3 ns **(D)** 4 ns
Explanation:

The possibilities are

TLB Hit*Cache Hit +

TLB Hit*Cache Miss +

TLB Miss*Cache Hit +

TLB Miss*Cache Miss

= 0.96*0.9*2 + 0.96*0.1*12 + 0.04*0.9*22 + 0,04*0.1*32

= 3.8

≈ 4

Why 22 and 32?
22 is because when TLB miss occurs it takes 1ns and the for the physical address it has to go through two level

page tables which are in main memory and takes 2 memory access and the that page is found in cache taking 1 ns which gives a total of 22

190.A processor uses 2-level page tables for virtual to physical address translation. Page tables for both levels are stored in the main memory. Virtual and physical addresses are both 32 bits wide. The memory is byte addressable. For virtual to physical address translation, the 10 most significant bits of the virtual address are used as index into the first level page table while the next 10 bits are used as index into the second level page table. The 12 least significant bits of the virtual address are used as offset within the page. Assume that the page table entries in both levels of page tables are 4 bytes wide. Further, the processor has a translation look-aside buffer (TLB), with a hit rate of 96%. The TLB caches recently used virtual page numbers and the corresponding physical page numbers. The processor also has a physically addressed cache with a hit rate of 90%. Main memory access time is 10 ns, cache access time is 1 ns, and TLB access time is also 1 ns.Suppose a process has only the following pages in its virtual address space: two contiguous code pages starting at virtual address 0x00000000, two contiguous data pages starting at virtual address 0×00400000, and a stack page starting at virtual address 0×FFFFF000. The amount of memory required for storing the page tables of this process is: [C]

(A) 8 KB **(B)** 12 KB **(C)** 16 KB **(D)** 20 KB

Explanation:

Breakup of given addresses into bit form:-

32bits are broken up as 10bits (L2) | 10bits (L1) | 12bits (offset)

first code page:

0x00000000 = 0000 0000 00 | 00 0000 0000 | 0000 0000 0000

so next code page will start from

0x00001000 = 0000 0000 00 | 00 0000 0001 | 0000 0000 0000

first data page:

0x00400000 = 0000 0000 01 | 00 0000 0000 | 0000 0000 0000

so next data page will start from

0x00401000 = 0000 0000 01 | 00 0000 0001 | 0000 0000 0000

only one stack page:

0xFFFFF000 = 1111 1111 11 | 11 1111 1111 | 0000 0000 0000

Now, for second level page table, we will just require 1 Page which will contain following 3 distinct entries i.e. 0000 0000 00,

0000 0000 01, 1111 1111 11.

Now, for each of these distinct entries, we will have 1-1 page in Level-1.Hence, we will have in total 4 pages and page size = 2^12 = 4KB.Therefore, Memory required to store page table = 4*4KB = 16KB.

191.Which of the following is not a form of memory? [C]

(A) instruction cache **(B)** instruction register

(C) instruction opcode **(D)** translation lookaside buffer

Explanation: Instruction Cache – Used for storing instructions that are frequently used
Instruction Register – Part of CPU's control unit that stores the instruction currently being executed.Instruction Opcode – It is the portion of a machine language instruction that specifies the operation to be performed.Translation Lookaside Buffer – It is a memory cache that stores recent translations of virtual memory to physical addresses for faster access

The optimal page replacement algorithm will select the page that [B]
(A) Has not been used for the longest time in the past.
(B) Will not be used for the longest time in the future.
(C) Has been used least number of times. **(D)** Has been used most number of times.
Explanation: The optimal page replacement algorithm will select the page whose next occurrence will be after the longest time in future. For example, if we need to swap a page and there are two options from which we can swap, say one would be used after 10s and the other after 5s, then the algorithm will swap out the page that would be required 10s later.

192.Dynamic linking can cause security concerns because: [B]
(A) Security is dynamic **(B)** The path for searching dynamic libraries is not known till runtime
(C) Linking is insecure **(D)** Crytographic procedures are not available for dynamic linking
Explanation: Static Linking and Static Libraries is the result of the linker making copy of all used library functions to the executable file. Static Linking creates larger binary files, and need more space on disk and main memory. Examples of static libraries (libraries which are statically linked) are, ***.a*** files in Linux and ***.lib*** files in Windows.
Dynamic linking and Dynamic Libraries Dynamic Linking doesn't require the code to be copied, it is done by just placing name of the library in the binary file. The actual linking happens when the program is run, when both the binary file and the library are in memory. Examples of Dynamic libraries (libraries which are linked at run-time) are, ***.so*** in Linux and ***.dll*** in Windows.In Dynamic Linking,the path for searching dynamic libraries is not known till runtime.
193.Which of the following statements is false? [D]
(A) Virtual memory implements the translation of a program's address space into physical memory address space
(B) Virtual memory allows each program to exceed the size of the primary memory
(C) Virtual memory increases the degree of multiprogramming
(D) Virtual memory reduces the context switching overhead
194.The process of assigning load addresses to the various parts of the program and adjusting the code and date in the program to reflect the assigned addresses is called [C]
(A) Assembly **(B)** Parsing **(C)** Relocation **(D)** Symbol resolution
Explanation:
Relocation of code is the process done by the linker-loader when a program is copied from external storage into main memory.A linker relocates the code by searching files and libraries to replace symbolic references of libraries with actual usable addresses in memory before running a program.Thus, option (C) is the answer.

195.Where does the swap space reside? [B]
(A) RAM **(B)** Disk **(C)** ROM **(D)** On-chip cache
Explanation: Swap space is an area on disk that temporarily holds a process memory image. When memory is full and process needs memory, inactive parts of process are put in swap space of disk.
196.Consider a virtual memory system with FIFO page replacement policy. For an arbitrary page access pattern, increasing the number of page frames in main memory will [C]
(A) always decrease the number of page faults **(B)** always increase the number of page faults
(C) sometimes increase the number of page faults **(D)** never affect the number of page faults

197.Consider a machine with 64 MB physical memory and a 32-bit virtual address space. If the page size is 4KB, what is the approximate size of the page table? [C]
(A) 16 MB **(B)** 8 MB **(C)** 2 MB **(D)** 24 MB
198.Suppose the time to service a page fault is on the average 10 milliseconds, while a memory access takes 1 microsecond. Then a 99.99% hit ratio results in average memory access time of (GATE CS 2000) [D]

(A) 1.9999 milliseconds **(B)** 1 millisecond **(C)** 9.999 microseconds **(D)** 1.9999 microseconds
Explanation: If any page request comes it will first search into page table, if present, then it will directly fetch the page from memory, thus in this case time requires will be only memory access time.But if required page

will not be found, first we have to bring it out and then go for memory access. This extra time is called page fault service time.Let hit ratio be p , memory access time be t1 , and page fault service time be t2.
Hence, average memory access time = p*t1 + (1-p)*t2

=(99.99*1 + 0.01*(10*1000 + 1))/100

=1.9999 *10^-6 sec

199.Consider a system with byte-addressable memory, 32 bit logical addresses, 4 kilobyte page size and page table entries of 4 bytes each. The size of the page table in the system in megabytes is _____ [B]
(A) 2 **(B)** 4 **(C)** 8 **(D)** 16
Explanation:
Number of entries in page table = 2^{32} / 4Kbyte
200.A computer system implements a 40 bit virtual address, page size of 8 kilobytes, and a 128-entry translation look-aside buffer (TLB) organized into 32 sets each having four ways. Assume that the TLB tag does not store any process id. Theminimum length of the TLB tag in bits is [D]
(A) 20 **(B)** 10 **(C)** 11 **(D)** 22
Explanation:
Total virtual address size = 40 Since there are 32 sets, set offset = 5

Since page size is 8kilobytes, word offset = 13 Minimum tag size = 40 - 5- 13 = 22

201.Consider six memory partitions of size 200 KB, 400 KB, 600 KB, 500 KB, 300 KB, and 250 KB, where KB refers to kilobyte. These partitions need to be allotted to four processes of sizes 357 KB, 210 KB, 468 KB and 491 KB in that order. If the best fit algorithm is used, which partitions are NOT allotted to any process?

(A) 200 KB and 300 KB **(B)** 200 KB and 250 KB
(C) 250 KB and 300 KB **(D)** 300 KB and 400 KB
Explanation: Best fit allocates the smallest block among those that are large enough for the new process. So the memory blocks are allocated in below order.
357 ---> 400

210 ---> 250

468 ---> 500

491 ---> 600

Sot the remaining blocks are of 200 KB and 300 KB

202.A Computer system implements 8 kilobyte pages and a 32-bit physical address space. Each page table entry contains a valid bit, a dirty bit three permission bits, and the translation. If the maximum size of the page table of a process is 24 megabytes, the length of the virtual address supported by the system is _____ bits [A]
(A) 36 **(B)** 32 **(C)** 28 **(D)** 40
Explanation:
Max size of virtual address can be calculated by calculating maximum number of page table entries.Maximum Number of page table entries can be calculated using given maximum page table size and size of a page table entry.Given maximum page table size = 24 MB.Let us calculate size of a page table entry.A page table entry has following number of bits.1 (valid bit) + 1 (dirty bit) + 3 (permission bits) + x bits to store physical address space of a page.

Value of x = (Total bits in physical address) - (Total bits for addressing within a page)

Since size of a page is 8 kilobytes, total bits needed withina page is 13.

So value of x = 32 - 13 = 19.

Putting value of x, we get size of a page table entry =1 + 1 + 3 + 19 = 24bits.

Number of page table entries

= (Page Table Size) / (An entry size)

= (24 megabytes / 24 bits)

= 2^{23}

Vrtual address Size

= (Number of page table entries) * (Page Size)

= 2^{23} * 8 kilobits

= 2^{36}

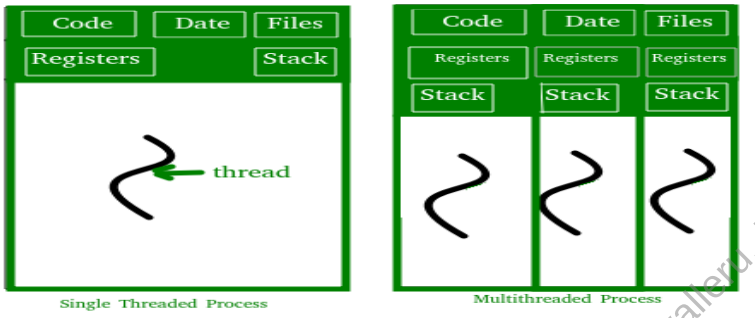
Therefore, length of virtual address space = 36

203.Which one of the following is NOT shared by the threads of the same process? [A]

(A) Stack **(B)** Address Space **(C)** File Descriptor Table **(D)** Message Queue

Explanation:

Threads can not share stack (used for maintaining function calls) as they may have their individual function call sequence.



204.Consider a fully associative cache with 8 cache blocks (numbered 0-7) and the following sequence of memory block requests:

4, 3, 25, 8, 19, 6, 25, 8, 16, 35, 45, 22, 8, 3, 16, 25, 7

If LRU replacement policy is used, which cache block will have memory block 7? [B]

(A) 4 **(B)** 5 **(C)** 6 **(D)** 7

Explanation: Block size is =8

Given 4, 3, 25, 8, 19, 6, 25, 8, 16, 35, 45, 22, 8, 3, 16, 25, 7

205.The storage area of a disk has innermost diameter of 10 cm and outermost diameter of 20 cm. The maximum storage density of the disk is 1400bits/cm. The disk rotates at a speed of 4200 RPM. The main memory of a computer has 64-bit word length and 1µs cycle time. If cycle stealing is used for data transfer from the disk, the percentage of memory cycles stolen for transferring one word is [C]

(A) 0.5% **(B)** 1% **(C)** 5% **(D)** 10%

Explanation:

Inner most diameter = 10 cm

Storage density = 1400 bits/cm

Capacity of each track :

= 3.14 * diameter * density

= 3.14 * 10 * 1400

= 43960 bits

Rotational latency = 60/4200 =1/70 seconds

It is given that the main memory of a computer has 64-bit word length and 1µs cycle time.
Data transferred in 1 sec = $64 * 10^6$ bits
Data read by disk in 1 sec = $43960 * 70 = 3.08 * 10^6$ bits
Total memory cycle = $(3.08 * 10^6) / (64 * 10^6) = 5\%$
Thus, option (C) is correct.

206.In a virtual memory system, size of virtual address is 32-bit, size of physical address is 30-bit, page size is 4 Kbyte and size of each page table entry is 32-bit. The main memory is byte addressable. Which one of the following is the maximum number of bits that can be used for storing protection and other information in each page table entry? [D]

(A) 2 **(B)** 10 **(C)** 12 **(D)** 14

Explanation:
Virtual memory = 2^{32} bytes
Physical memory = 2^{30} bytes
Page size = Frame size = $4 * 10^3$ bytes = $2^2 * 2^{10}$ bytes = 2^{12} bytes
Number of frames = Physical memory / Frame size = $2^{30}/2^{12} = 2^{18}$
Therefore, Numbers of bits for frame = 18 bits
Page Table Entry Size = Number of bits for frame + Other information
Other information = 32 – 18 = 14 bits

207.In a particular Unix OS, each data block is of size 1024 bytes, each node has 10 direct data block addresses and three additional addresses: one for single indirect block, one for double indirect block and one for triple indirect block. Also, each block can contain addresses for 128 blocks. Which one of the following is approximately the maximum size of a file in the file system? [B]

(A) 512 MB **(B)** 2GB **(C)** 8GB **(D)** 16G

Explanation:
Maximum size of the File System = Summation of size of all the data blocks whose addresses belongs to the file.

Given:Size of 1 data block = 1024 Bytes

No. of addresses which 1 data block can contain = 128

Now, Maximum File Size can be calculated as:

10 direct addresses of data blocks = 10*1024

1 single indirect data block = 128*1024

1 doubly indirect data block = 128*128*1024

1 triple indirect data block = 128*128*128*1024

Hence,Max File Size = 10*1024 + 128*1024 + 128*128*1024 +
128*128*128*1024 Bytes
= 2113674*1024 Bytes
= 2.0157 GB ~ 2GB

208.A two-way switch has three terminals a, b and c. In ON position (logic value 1), a is connected to b, and in OFF position, a is connected to c. Two of these two-way switches S1 and S2 are connected to a bulb as shown below. Which of the following expressions, if true, will always result in the lighting of the bulb ? [C]

(A) S1.S2' **(B)** S1+S2 **(C)** $(S1 \oplus S2)'$ **(D)** $S1 \oplus S2$

Explanation: If we draw truth table of the above circuit,it'll be

S1	S2	Bulb
0	0	On
0	1	Off

1 0 Off
1 1 On
= (S1⊕S2)'

209.Consider a 2-way set associative cache memory with 4 sets and total 8 cache blocks (0-7) and a main memory with 128 blocks (0-127). What memory blocks will be present in the cache after the following sequence of memory block references if LRU policy is used for cache block replacement. Assuming that initially the cache did not have any memory block from the current job? [C]
0 5 3 9 7 0 16 55

(A) 0 3 5 7 16 55 **(B)** 0 3 5 7 9 16 55 **(C)** 0 5 7 9 16 55 **(D)** 3 5 7 9 16 55

Explanation:

2-way set associative cache memory, .i.e K = 2.

No of sets is given as 4, i.e. S = 4 (numbered 0 - 3)

No of blocks in cache memory is given as 8, i.e. N =8 (numbered from 0 -7)

Each set in cache memory contains 2 blocks.The number of blocks in the main memory is 128, i.e M = 128. (numbered from 0 -127).A referred block numbered X of the main memory is placed in the set numbered (X mod S) of the the cache memory. In that set, the block can be placed at any location, but if the set has already become full, then the current referred block of the main memory should replacea block in that set according to some replacement policy. **Here the replacement policy is LRU (i.e. Least Recently Used block should be replaced with currently referred block).**X (Referred block no) and the corresponding Set values are as follows:

X-->set no (*X mod 4*)
0--->0 (block 0 is placed in set 0, set 0 has 2 empty block locations,
block 0 is placed in any one of them)
5--->1 (block 5 is placed in set 1, set 1 has 2 empty block locations,
block 5 is placed in any one of them)
3--->3 (block 3 is placed in set 3, set 3 has 2 empty block locations,
block 3 is placed in any one of them)
9--->1 (block 9 is placed in set 1, set 1 has currently 1 empty block location,
block 9 is placed in that, now set 1 is full, and block 5 is the
least recently used block)
7--->3 (block 7 is placed in set 3, set 3 has 1 empty block location,
block 7 is placed in that, set 3 is full now,
and block 3 is the least recently used block)
0--->block 0 is referred again, and it is present in the cache memory in set 0,
so no need to put again this block into the cache memory.
16--->0 (block 16 is placed in set 0, set 0 has 1 empty block location,
block 0 is placed in that, set 0 is full now, and block 0 is the LRU one)
55--->3 (block 55 should be placed in set 3, but set 3 is full with block 3 and 7,
hence need to replace one block with block 55, as block 3 is the least
recently used block in the set 3, it is replaced with block 55.

Hence the main memory blocks present in the cache memory are : **0, 5, 7, 9, 16, 55 .**
(**Note**: block 3 is not present in the cache memory, it was replaced with block 55)

210.A disk has 8 equidistant tracks. The diameters of the innermost and outermost tracks are 1 cm and 8 cm respectively. The innermost track has a storage capacity of 10 MB.What is the total amount of data that can be stored on the disk if it is used with a drive that rotates it with (i) Constant Linear Velocity (ii) Constant Angular Velocity? [D]

(A) (i) 80 MB (ii) 2040 MB **(B)** (i) 2040 MB (ii) 80 MB **(C)** (i) 80 MB (ii) 360 MB **(D)** (i) 360 MB (ii) 80 MB

Explanation:

Constant linear velocity :

Diameter of inner track = d = 1cm

Circumference of inner track :

= 2 * 3.14 * (d/2)

= 3.14 cm

Storage capacity = 10 MB (given)

Circumference of all equidistant tracks :

= 2 * 3.14 *(0.5 + 1 + 1.5 + 2 + 2.5 + 3+ 3.5 + 4)

= 113.14cm

Here, 3.14 cm holds 10 MB.Therefore, 1 cm holds 3.18 MB.

113.14 cm holds 113.14 * 3.18 = 360 MB.

Total amount of data that can be stored on the disk = 360 MB

Constant angular velocity : In case of CAV, the disk rotates at a constant angular speed.

Same rotation time is taken by all the tracks.Total amount of data that can be stored on the disk = 8 * 10 = 80 MB.Thus, option (D) is correct.

211.Consider a computer system with 40-bit virtual addressing and page size of sixteen kilobytes. If the computer system has a one-level page table per process and each page table entry requires 48 bits, then the size of the per-process page table is _____megabytes. [A]

Note : This question was asked as Numerical Answer Type.

(A) 384 **(B)** 48 **(C)** 192 **(D)** 96

Explanation: Size of memory = 2^{40}

Page size = 16KB = 2^{14}

No of pages= size of Memory/ page size = 2^{40} / 2^{14} = 2^{26}

Size of page table = 2^{26} * 48/8 bytes = 2^{6}*6 MB =384 MB

212.Consider a computer system with ten physical page frames. The system is provided with an access sequence a1, a2, …, a20, a1, a2, …, a20), where each ai number. The difference in the number of page faults between the last-in-first-out page replacement policy and the optimal page replacement policy is _____ [B]

(A) 0 **(B)** 1 **(C)** 2 **(D)** 3

Explanation: LIFO stands for last in, first out

a1 to a10 will result in page faults, So 10 page faults from a1 to a10.

Then a11 will replace a10(last in is a10), a12 will replace a11 and so on till a20, so 10 page faults from a11 to a20 and a20 will be top of stack and a9…a1 are remained as such.

Then a1 to a9 are already there. So 0 page faults from a1 to a9.

a10 will replace a20, a11 will replace a10 and so on. So 11 page faults from a10 to a20. So total faults will be 10+10+11 = 31.

Optimal

a1 to a10 will result in page faults, So 10 page faults from a1 to a10.

Then a11 will replace a10 because among a1 to a10, a10 will be used later, a12 will replace a11 and so on. So 10 page faults from a11 to a20 and a20 will be top of stack and a9…a1 are remained as such.

Then a1 to a9 are already there. So 0 page faults from a1 to a9.

a10 will replace a1 because it will not be used afterwards and so on, a10 to a19 will have 10 page faults.

a20 is already there, so no page fault for a20.

Total faults 10+10+10 = 30.

Difference = 1

213.In which one of the following page replacement algorithms it is possible for the page fault rate to increase even when the number of allocated frames increases? [D]

(A) LRU (Least Recently Used) **(B)** OPT (Optimal Page Replacement)

(C) MRU (Most Recently Used) **(D)** FIFO (First In First Out)

Explanation: In some situations FIFO page replacement gives more page faults when increasing the number of

page frames. This situation is Belady's anomaly. **Belady's anomaly** proves that it is possible to have more page faults when increasing the number of page frames while using the First in First Out (FIFO) page replacement algorithm. For example, if we consider reference string 3 2 1 0 3 2 4 3 2 1 0 4 and 3 slots, we get 9 total page faults, but if we increase slots to 4, we get 10 page faults.

214.The address sequence generated by tracing a particular program executing in a pure demand paging system with 100 bytes per page is

0100, 0200, 0430, 0499, 0510, 0530, 0560, 0120, 0220, 0240, 0260, 0320, 0410.

Suppose that the memory can store only one page and if x is the address which causes a page fault then the bytes from addresses x to x + 99 are loaded on to the memory.

How many page faults will occur ? [C]

(A) 0 **(B)** 4 **(C)** 7 **(D)** 8

Explanation:

Address	**Page faults**	**last byte in memory**
0100	page fault,	199
0200	page fault,	299
0430	page fault,	529
0499	no page fault	
0510	no page fault	
0530	page fault,	629
0560	no page fault	
0120	page fault,	219
0220	page fault,	319
0240	no page fault	
0260	no page fault	
0320	page fault,	419
0410	no page fault	

So, 7 is the answer- (C)

215.A paging scheme uses a Translation Look-aside Buffer (TLB). A TLB-access takes 10 ns and a main memory access takes 50 ns. What is the effective access time(in ns) if the TLB hit ratio is 90% and there is no page-fault? [C]

(A) 54 **(B)** 60 **(C)** 65 **(D)** 75

Explanation: Effective access time = hit ratio * time during hit + miss ratio * time during miss

TLB time = 10ns,

Memory time = 50ns

Hit Ratio= 90%

E.A.T. = (0.90)*(60)+0.10*110=65

216.Assume that a main memory with only 4 pages, each of 16 bytes, is initially empty. The CPU generates the following sequence of virtual addresses and uses the Least Recently Used (LRU) page replacement policy.

0, 4, 8, 20, 24, 36, 44, 12, 68, 72, 80, 84, 28, 32, 88, 92

How many page faults does this sequence cause? What are the page numbers of the pages present in the main memory at the end of the sequence? [B]

(A) 6 and 1, 2, 3, 4 **(B)** 7 and 1, 2, 4, 5 **(C)** 8 and 1, 2, 4, 5 **(D)** 9 and 1, 2, 3, 5

217.Match the following flag bits used in the context of virtual memory management on the left side with the different purposes on the right side of the table below. [D]

Name of the bit		**Purpose**	
I.	Dirty	a.	Page initialization
II.	R/W	b.	Write-back policy
III.	Reference	c.	Page protection
IV.	Valid	d.	Page replacement policy

(A) I-d, II-a, III-b, IV-c **(B)** I-b, II-c, III-a, IV-d **(C)** I-c, II-d, III-a, IV-b **(D)** I-b, II-c, III-d, IV-a

Explanation: Dirty:- Dirty bit is used for whenever a page has been modify and we try to replace it and then this has to be written back so it write-back or leave it. Therefore dirty bit is used for write-back policy.

R/W:- R/W is used for page protection.

Reference:- It says which page has not referred recently. Hence it is used for page replacement policy.

Valid:- It says wether the page page you are looking for it present in MM or not. If not present in MM then load into the SM i.e. called page intiallisation.

218.Assume GeeksforGeeks implemented the new page replacement algorithm in virtual memory and given its name as 'Geek'. Consider the working strategy of Geek as following-

- Each page in memory maintains a count which is incremented if the page is referred and no page fault occurs.
- If a page fault occurs, the physical page with zero count or smallest count is replaced by new page and if more than one page with zero count or smallest count then it uses FIFO strategy to replace the page.

Find the number of page faults using Geeks algorithm for the following reference string (assume three physical frames are available which are initially free) [C]

Reference String : "A B C D A B E A B C D E B A D"

(A) 7 **(B)** 9 **(C)** 11 **(D)** 13

Explanation:

A_0	D_0	D_0	D_0	E_0	D_0	C_0	D_0	E_0	E_0	D_0
B_0	B_0	A_0	A_0	A_0	A_1	A_1	A_1	A_1	A_2	A_2
C_0	C_0	C_0	B_0	B_0	B_1	B_1	B_1	B_1	B_2	B_2
Page faults	3	1	1	1	1	0	1	1	1	0	1

(If page fault, count = 0. Otherwiseincrement count for referred page.)

11 page faults.

219.If LRU and Geek page replacement are compared (in terms of page faults) only for above reference string then find the correct statement from the following: [C]

(A) LRU and Geek are same **(B)** LRU is better than Geek **(C)** Geek is better than LRU **(D)** None

A	D	D	D	E	C	C	C	B	B	B
B	B	A	A	A	A	D	D	D	A	A
C	C	C	B	B	B	B	E	E	E	D
Page faults	3	1	1	1	1	1	1	1	1	1	1

13 page faults

Geek is better then LRU for these references. Hence (C) is correct option.

Explanation

220.Listed below are some operating system abstractions (in the left column) and the hardware components or mechanism (in the right column) that they are abstractions of. Which of the following matching of pairs is correct? [C]

A. Thread	1. Interrupt
B. Virtual address space	2. Memory
C. File system	3. CPU
D. Signal	4. Disk

(A) A-2, B-4, C-3, D-1 **(B)** A-1, B-2, C-3, D-4 **(C)** A-3, B-2, C-4, D-1 **(D)** A-4, B-1, C-2, D-3

Explanation:

1. Thread is handled by CPU.
2. Virtual address space (it is a set of virtual memory addresses that a process can use) is associated with memory management.
3. File System is used for disk management. 4.Interrupt is a type of signal.

221.A multi-user, multi-processing operating system cannot be implemented on hardware that does not support: [C]

a) Address translation b) DMA for disk transfer

c) At least two modes of CPU execution (privileged and non-privileged). d) Demand Paging

(A) Only A **(B)** Both A, B **(C)** A, B and C **(D)** A, B and D

Explanation: The correct answer would be (a) and (c) as address translation is required in the multi-programming so that no process can go into any other process's memory. And atleast 2 modes should be present in the CPU execution so that the privileged mode could control the resource allocation of the unprivileged mode users. The DMA and Demand Paging improves the performance of the OS. Hence they are not necessary conditions in a multi-programming. But since (a) and (c) is not answers mentions in the options so the next best option will be (C) containing both (a) and (c) along with (b).

So, option (C) is correct.

222.Which of the following is/are advantages of virtual memory? [C]

a) Faster access to memory on an average. b) Processes can be given protected address spaces.

c) Linker can assign addresses independent of where the program will be loaded in physical memory.

d) Programs larger than the physical memory size can be run.

(A) a and b **(B)** b and c **(C)** b and d **(D)** All of the above

Explanation: Virtual memory provides an interface through which processes access the physical memory. So,

A. Swapping of pages increases time while virtual memory concept is there and it slower the execution as compared to Direct access from physical memory. So, it is false.

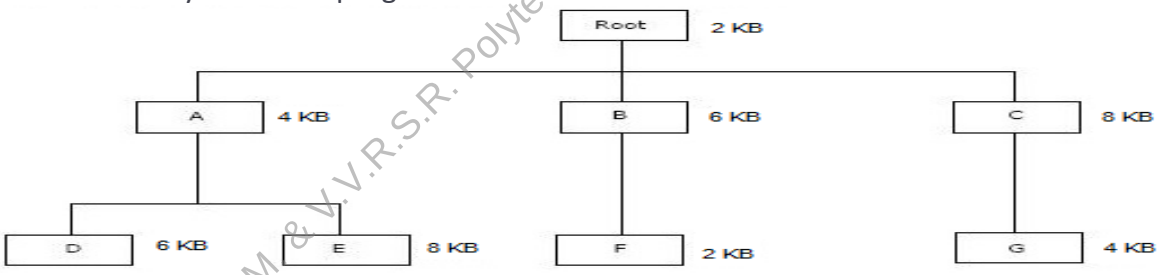
B. Without virtual memory, it is difficult to give protected address space to processes as they will be accessing physical memory directly. So, it is true

C. Some other method can be used to assign addresses independent of where the program. so it is false.

D. Virtual memory allows a process to run using a virtual address space which is larger in size then the address of physical memory and pages are swapped in between physical memory and virtual memory when physical memory gets full. So, it is True.

So, statement (b) and (d) are Correct.

223.The overlay tree for a program is as shown below:



What will be the size of the partition (in physical memory) required to load (and run) this program? [B]

(A) 12 KB **(B)** 14 KB **(C)** 10 KB **(D)** 8 KB

Explanation: Depth First Traversal from left to right. when we visit a node for first time, it is loaded in main memory and when we visit it for last time, it is pulled out.

For the above program, maximum memory will be required when running code portion present at leaves.

For D = 2+4+6 = 12

For E = 2+4+8 = 14

For F = 2+6+2 = 10

For G = 2+8+4 = 14

Max(12, 14, 10, 14) = 14

Free disk space can be kept track of using a free list or a bit map. Disk addresses require d bits. For a disk with 13 blocks, F of which is free, state the condition under which the free list uses less space than the bit map.

225.Thrashing [C]

(A) reduces page I/O **(B)** decreases the degree of multiprogramming
(C) implies excessive page I/O **(D)** improves the system performance

226.Dirty bit for a page in a page table [A]

(A) helps avoid unnecessary writes on a paging device **(B)** helps maintain LRU information
(C) allows only read on a page **(D)** none of the above

Explanation: Dirty bit is used whenever a page has been modifying and we are trying to replace it and then this has to be written back so wether to do write-back or leave it dirty bit is required so dirty bit used for write-back. Hence, Dirty bit for a page in a page table helps avoid unnecessary writes on a paging device

227.A 1000 Kbyte memory is managed using variable partitions but no compaction. It currently has two partitions of sizes 200 Kbytes and 260 Kbytes respectively. The smallest allocation request in Kbytes that could be denied is for [B]

(A) 151 **(B)** 181 **(C)** 231 **(D)** 541

228.Consider 4-way set-associative cache memory has the following properties:

- Data words are 32 bits each
- A cache block will contain 2048 bits of data
- The address supplied from the CPU is 32 bits long
- There are 2048 blocks in the cache
- Addresses are to the word

Number of bits in tag? [A]

(A) 17 **(B)** 15 **(C)** 16 **(D)** None of the above

Explanation: – There are still 6 bits in the offset; data is word addressed
Number of bits in index?
– We now need one less bit of index because we address to the set
→ 2^11 blocks / 2^2 blocks/set = 2^9 sets
→ (9 bits of index needed)
Number of bits in tag?
→ 32 – 6 – 9 = 17 bits.
So, option (A) is correct.

230.Recall that Belady's anomaly is that the pages-fault rate may increase as the number of allocated frames increases and also recall convey effect where first process can increase waiting time for all processes. Now consider the following statements:

S1 : Random page replacement algorithm (where
a page chosen at random is replaced)
suffers from Belady's anomaly.

S2 : Random CPU scheduling algorithm
suffers from convoy effect.

Which of the following is CORRECT? [A]

(A) S1 is true, S2 is true **(B)** S1 is true, S2 is false **(C)** S1 is false , S2 is true **(D)** S1 is false, S2 is false

Explanation: Both statements can be true, because Random page replacement algorithm may behave like FIFO page replacement algorithm and Random CPU scheduling algorithm may behave like FCFS CPU scheduling algorithm.
Option (A) is correct.

231Which of the following option is false? [A]

(A) Internal fragmentation occurs when total memory space exists to satisfy the a request, but it is not contiguous. External fragmentation occurs when allocated memory may be slightly larger the requested memory, this size difference is memory internal to a partition,but not being used.

(B) Pure paging does not have external fragmentation, but may have internal fragmentation and Pure segmentation may have external fragmentation.

(C) Both (A) and (B) **(D)** Neither (A) nor (B)

Explanation: External fragmentation occurs when total memory space exists to satisfy a request, but it is not contiguous. Internal fragmentation occurs when allocated memory may be slightly larger the requested memory, this size difference is memory internal to a partition,but not being used.

Pure paging does not have external fragmentation, but may have internal fragmentation and Pure segmentation may have external fragmentation.Option (A) is false.

232.Consider a process executing on an operating system that uses demand paging. The average time for a memory access in the system is M units if the corresponding memory page is available in memory, and D units if the memory access causes a page fault. It has been experimental measured that the average time taken for a memory access in the process is X units.Which one of the following is the correct expression for the page fault rate experienced by the process? [B]

(A) (D – M) / (X – M) **(B)** (X – M) / (D – M) **(C)** (D – X) / (D – M) **(D)** (X – M) / (D – X)

Explanation: Given, average time for a memory access = M units if page hits, and average time for a memory access = D units if page fault occurred.And total/experimental average time taken for a memory access = X units.Let page fault rate is p. Therefore,

Average memory access time = (1 – page fault rate) * memory access time when no page fault + Page fault rate * Memory access time when page fault

→ X = (1 – p)*M + p*D = M – M*p + p*D

→ X = M + p(D – M)

→ (X – M) = p(D – M)

→ p = (X – M) / (D – M)

So, option (B) is correct.

233.In designing a computer's cache system, the cache block (or cache line) size is an important parameter. Which one of the following statements is correct in this context? [A]

(A) Smaller block size incurs lower cache miss penalty **(B)** Smaller block size implies better spatial locality

(C) Smaller block size implies smaller cache tag **(D)** Smaller block size implies lower cache hit time

Explanation:

234A counting semaphore was initialised to 7. Then 20 P (wait) operations and x V (signal) operations were completed on this semaphore. If the final value of semaphore is 5, then the value x will be [C]

(A) 0 **(B)** 13 **(C)** 18 **(D)** 5

Explanation: In semaphores, P operation means wait operation and it decrements the value of the counting semaphore by 1 whereas V operation means signal operation and it increments the value of the counting semaphore..

Here a counting semaphore S = 7

After 20 P operations , s = 7-20 = -13

After 'x' V operations value of S=5, So -13 + xV = 5 and S = 18 .

235.Consider a virtual page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1 Suppose a demand paged virtual memory system running on a computer system such that the main memory has 3 page frames. Then _____ page replacement algorithm has minimum number of page faults. [D]

(A) FIFO **(B)** LIFO **(C)** LRU **(D)** Optimal

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
			1	1	1	1	0	0	0	3	3	3	3	3	2	2	2	2	2	1
		0	0	0	0	3	3	3	2	2	2	2	2	1	1	1	1	1	0	0
	7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
	F	F	F	F		F	F	F	F	F	F			F	F			F	F	F

Explanation:

1. In FIFO algorithm 15 page fault will occur.

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
		1	2	2	3	3	4	2	3	3	3	2	1	2	2	1	1	1	1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
F	F	F	F		F		F	F	F			F	F	F					

2. In LIFO algorithm 11 page fault will occur.

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
		1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
	0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
F	F	F	F		F		F	F	F	F			F	F			F		

3. In LRU algorithm 12 page fault will occur.

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
		1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
	0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
7	7	7	2	2	2	2	2	2	2	2	2	2	2	2	2	2	7	7	7
F	F	F	F		F		F			F			F				F		

4. In optimal algorithm 9 page fault will occur.

236.Consider a system with five processes P_0 through P_4 and three resource types A, B and C. Resource type A has seven instances, resource type B has two instances and resource type C has six instances suppose at time T_0we have the following allocation.

Process	Allocation			Request			Available		
	A	B	C	A	B	C	A	B	C
P_0	0	1	0	0	0	0	0	0	0
P_1	2	0	0	2	0	2			
P_2	3	0	3	0	0	0			
P_3	2	1	1	1	0	0			
P_4	0	2	2	0	0	2			

If we implement Deadlock detection algorithm we claim that system is _____. [D]

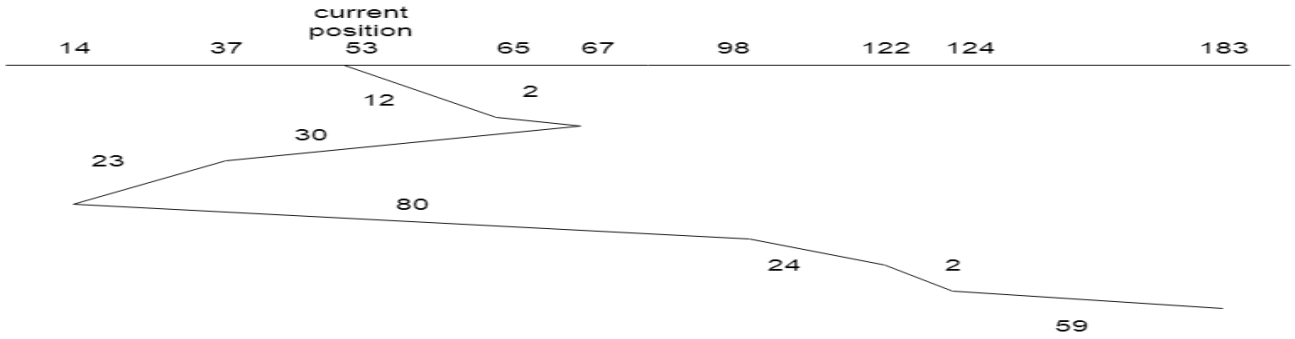
(1) Semaphore (2) Deadlock state (3) Circular wait (4) Not in deadlock state

(A) (1), (2) and (3) **(B)** (1) and (3) **(C)** (3) and (4) **(D)** Only option (4)

237.Consider a disk queue with requests for I/O to blocks on cylinders 98, 183, 37, 122, 14, 124, 65, 67. Suppose SSTF disk scheduling algorithm implemented to meet the requests then the total number of head movements are _____ if the disk head is initially at 53.

(A) 224 **(B)** 248 **(C)** 236 **(D)** 240

Explanation:



Now add all seek time:

12+2+30+23+84+24+2+59=236

So, option (C) is correct.

238.In a paging system, it takes 30 ns to search translation Look-a-side Buffer (TLB) and 90 ns to access the main memory. If the TLB hit ratio is 70%, the effective memory access time is :[B]

(A) 48ns **(B)** 147ns **(C)** 120ns **(D)** 84ns

Explanation:

Effective access time = Hit ratio*Time during hit + Miss Ratio * Time During Miss

=0.7*(30+90) + 0.3 (30+90+90)

=84 + 63

=147 ns

So, option (B) is correct.

239.Which of the following is incorrect for virtual memory? [B]

(A) Large programs can be written **(B)** More I/O is required

(C) More addressable memory available **(D)** Faster and easy swapping of process

Explanation:

1. We can write large program when we have memory extension.
2. Less I/O requirement in case of virtual memory.
3. More addressable memory available with virtual memory
4. Faster and easy swapping of process with the help of page replacement policy in virtual memory.So, option (B) is correct.

240.A CPU generates 32-bit virtual addresses. The page size is 4 KB. The processor has a translation look-aside buffer (TLB) which can hold a total of 128 page table entries and is 4-way set associative. The minimum size of the TLB tag is: [C]

(A) 11 bits **(B)** 13 bits **(C)** 15 bits **(D)** 20 bits

241.In which one of the following page replacement policies, Belady's anomaly may occur? [A]

(A) FIFO **(B)** Optimal **(C)** LRU **(D)** MRU

Explanation: Belady's Anomaly is a situation in which increasing the number of frames allocated to a process results in increased number of page faults. This usually happens in FIFO page replacement algorithm. The anomaly occurs when the page replacement algorithm will remove a page that will be needed in the immediate future.

242.A memory management system has 64 pages with 512 bytes page size. Physical memory consists of 32 page frames. Number of bits required in logical and physical address are respectively: [C]

(A) 14 and 15 **(B)** 14 and 29 **(C)** 15 and 14 **(D)** 16 and 32

Explanation:

we know that Number of pages = virtual memory space / page size.

and Number of frames = physical memory space / frame size.

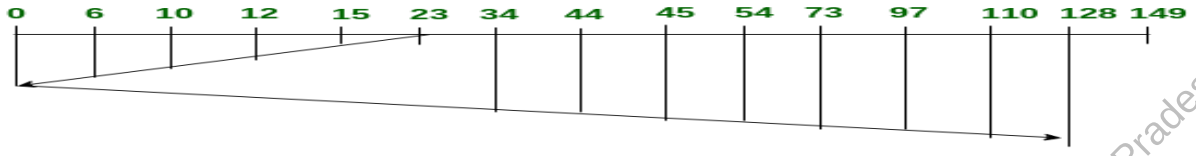
and page size is equal to frame size.

According to question and given data: virtual memory space = Number of pages * page size

i.e. virtual memory space = 64 * 512 B. virtual memory space = $2^6 * 2^9$B i.e. = 2^{15}B.

So, 15 bits are required for virtual memory space.physical memory space = Number of frames * frame size.physical memory space = 32 * 512 B. physical memory space = $2^5 * 2^9$B.i.e. = 2^{14}B.
So, 14 bits are required for virtual memory space.So, option (C) is correct.
243.Consider a disk queue with I/O requests on the following cylinders in their arriving order:
6, 10, 12, 54, 97, 73, 128, 15, 44, 110, 34, 45

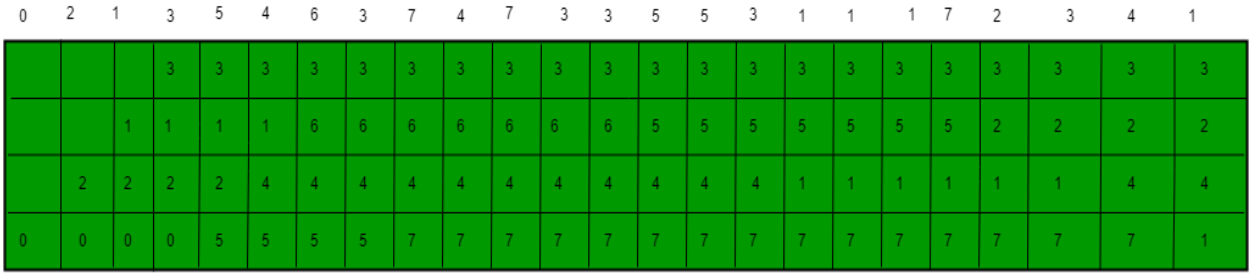
The disk head is assumed to be at cylinder 23 and moving in the direction of decreasing number of cylinders. Total number of cylinders in the disk is 150. The disk head movement using SCAN-scheduling algorithm is:[d]
(A) 172 **(B)** 173 **(C)** 227 **(D)** None
Explanation: According to SCAN disk scheduling algorithm:



When it completed its last given sequence from input, it stops.
So, total disk head movement = 23+128 = 151.

244.If there are 32 segments, each size 1 k bytes, then the logical address should have [c]
(A) 13 bits **(B)** 14 bits **(C)** 15 bits **(D)** 16 bits
Explanation: Number of bits required for 32 segments, each of 1 k bytes
$= 2^5 * 2^{10}$
$= 2^{15}$
So, 15 bits are required.
245.At a particular time of computation the value of a counting semaphore is 7. Then 20 P operations and 15 V operations were completed on this semaphore. The resulting value of the semaphore is : [B]
(A) 42 **(B)** 2 **(C)** 7 **(D)** 12
Explanation: Value of a counting semaphore = 7
After 20 P operations value of semaphore = 7 – 20 = -13
After 15 V operations value of semaphore = -13 + 15 = 2
So, option (B) is correct.
246.Increasing the RAM of a computer typically improves performance because: [C]
(A) Virtual Memory increases **(B)** Larger RAMs are faster
(C) Fewer page faults occur **(D)** Fewer segmentation faults occur
247.Dirty bit for a page in a page table [A]
(A) helps avoid unnecessary writes on a paging device **(B)** helps maintain LRU information
(C) allows only read on a page **(D)** None of the above
Explanation: Dirty bit is used to keep the track whether a page in cache memory is modified or not. Dirty bit is set when the page is modified and needs to be copied back to main memory. So it is used to avoid unnecessary writes on a paging device.
248.Please comment below if you find anything wrong in the above post

Suppose that the virtual Address space has eight pages and physical memory with four page frames. If LRU page replacement algorithm is used, _____ number of page faults occur with the reference string. [A]
0 2 1 3 5 4 6 3 7 4 7 3 3 5 5 3 1 1 1 7 2 3 4 1
(A) 13 **(B)** 12 **(C)** 11 **(D)** 10



Explanation:

So, option (A) is correct.

249.Consider the following segment table in segmentation scheme:

SegmentID	Base	Limit
0	200	200
1	500	12510
2	1527	498
3	2500	50

What happens if the logical address requested is -Segment Id 2 and offset 1000? [B]

(A) Fetches the entry at the physical address 2527 for segment Id2

(B) A trap is generated **(C)** Deadlock **(D)** Fetches the entry at offset 27 in Segment Id 3

Explanation: Segment-2 has a base address = 1527 and limit address = 498, So it can only access the memory location from 1527 till 1527 + 498, if the process tries to access the memory with offset 1000 then a segmentation fault trap will be generated.

250.Using the page table shown below, translate the physical address 25 to virtual address. The address length is 16 bits and page size is 2048 words while the size of the physical memory is four frames. [D]

Page	Present(1-In 0-out)	Frame
0	1	3
1	1	2
2	1	0
3	0	-

(A) 25 **(B)** 6169 **(C)** 2073 **(D)** 4121

Explanation: Given, virtual address size is 16 bits and page size is 2^{11} bytes. Therefore,

Number of pages = $2^{16}/2^{11} = 2^5$

So, physical address = (number of frames) * (size of each frame) = $4 * 2^{11} = 2^{13}$

Physical address $(25)_{10} = (0000000011001)_2$ in 13 bits

Here first two bits represent frame number and another 11 bits represent the offset within the page: $(00\ 00000011001)_2$

According to the page table: frame 00 is mapped on with page number 2

Page # 2 = $(00010)_2$

Offset within the page = $(00000011001)_2$

Therefore, 16 bit virtual address = $(00010\ 00000011001)_2 = (4121)_{10}$

So, option (D) is correct.

251.A computer has 16 pages of virtual address space but the size of main memory is only four frames. Initially the memory is empty. A program references the virtual pages in the order 0, 2, 4, 5, 2, 4, 3, 11, 2, 10. How many page faults occur if LRU page replacement algorithm is used? [C]

(A) 3 **(B)** 5 **(C)** 7 **(D)** 8

252.An operating system supports a paged virtual memory, using a central processor with a cycle time of one microsecond. It costs an additional one microsecond to access a page other than the current one. Pages have 1000 words, and the paging device is a drum that rotates at 3000 revolutions per minute and transfers one million words per second. Further, one percent of all instructions executed accessed a page other than the current page. The instruction that accessed another page, 80% accessed a page already in memory and when a new page was required, the replaced page was modified 50% of the time. What is the effective access time

on this system, assuming that the system is running only one process and the processor is idle during drum transfers? [B]

(A) 30 microseconds **(B)** 34 microseconds **(C)** 60 microseconds **(D)** 68 microseconds

Explanation: Effective access time = 0.99 * (1 µs) + 0.8 *.02 µs + 0.001 * (10000 µs + 1000 µs) + 0.001 * (20000 µs + 2000 µs)

= (0.99 +0.016+22.0+11.0) µs

= 34 µs

253.Consider the following page reference string :

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 2, 3, 7, 6, 3, 2, 1, 2, 3, 6

Which of the following options, gives the correct number of page faults related to LRU, FIFO, and optimal page replacement algorithms respectively, assuming 05 page frames and all frames are initially empty ? [B]

(A) 10, 14, 8 **(B)** 8, 10, 7 **(C)** 7, 10, 8 **(D)** 7, 10, 7

Explanation: In LRU:

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
						5	5	5	5	5	5	7	7	7	7	7	7	7	7
			4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3
		3	3	3	3	3	6	6	6	6	6	6	6	6	6	6	6	6	6
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
F	F	F	F			F	F				F	F							

In FIFO:

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
						5	5	5	5	5	5	7	7	7	7	7	7	7	7
			4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3
		3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	2	2
	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	6	6	6	6	6	6	6	6	6	6	6	6	6
F	F	F	F			F	F		F	F	F	F							

In Optimal:

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
						5	5	5	5	5	5	7	7	7	7	7	7	7	7
			4	4	4	4	6	6	6	6	6	6	6	6	6	6	6	6	6
		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	1	1	6	6	6	6	6	6	6
F	F	F	F			F	F					F							

Respectively there are 8, 10, 7 page faults.

So, option (B) is correct.

254.Dirty bit is used to indicate which of the following? [C]

(A) A page fault has occurred **(B)** A page has corrupted data

(C) A page has been modified after being loaded into cache **(D)** An illegal access of page

Explanation: When a page is modified inside the cache and the changes need to be stored back in the main memory, the valid bit is set to 1 so as to maintain the record of modified pages.

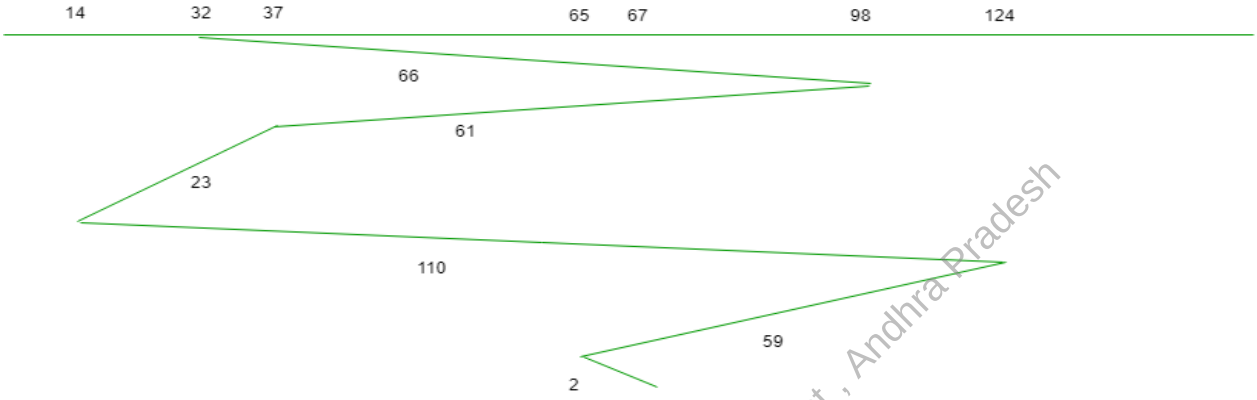
255.What is the size of the physical address space in a paging system which has a page table containing 64 entries of 11 bit each (including valid and invalid bit) and a page size of 512 bytes?

[C]

(A) 2^{11} **(B)** 2^{15} **(C)** 2^{19} **(D)** 2^{20}

Explanation: Size of Physical Address = Paging bits + Offset bits

Paging bits = 11 – 1 = 10 (As 1 valid bit is also included)
Offset bits = $\log_2$page size = 9
Size of Physical Address = 10 + 9 = 19 bits
Option (C) is correct.
256.If the Disk head is located initially at track 32, find the number of disk moves required with FCFS scheduling criteria if the disk queue of I/O blocks requests are: [C]
98, 37, 14, 124, 65, 67
(A) 320 **(B)** 322 **(C)** 321 **(D)** 319



Explanation:
Total disk movement = 66 + 61 + 23 + 110 + 2 = 321.
So, option (C) is correct.
257.Suppose that the time to do a null remote procedure call (RPC) (i.e. 0 data bytes) is 1.0 msec, with an additional 1.5 msec for every 1K of data. How long does it take to read 32 K from the file server as 32 1K RPCs ? [B]
(A) 49 msec **(B)** 80 msec **(C)** 48 msec **(D)** 100 msec
Explanation: According to given question
1K RPC takes – 1.5 * 32 + 1.0 = 49.0 msec
32 1K RPC will take:
1.5 * 32 + 1.0 * 32= 80.0 msec
258.Which of the following is/are likely the cause/s of thrashing? [B]
(A) Page size was very small **(B)** There are too many programs running in the system
(C) Least recently used policy is used for page replacement
(D) First in First out policy is used for page replacement
Explanation: When the degree of multiprogramming keep on increasing, the number of page faults increase due to the insufficient memory space per process and there's hardly any progress in process execution. This situation is termed as Thrashing.Clearly, Option (B) is correct.
259.Consider a logical address space of 8 pages of 1024 words each, mapped onto a physical memory of 32 frames. How many bits are there in the physical address and logical address respectively? [C]
(A) 5, 3 **(B)** 10, 10 **(C)** 15, 13 **(D)** 15, 15
Explanation: Logical address space has 8 pages, size of each page, offset = 1024 words
Number if bits in page# field of the logical address = $\log_2$8 bits = 3 bits
Offset bits = $\log_2$1024 = 10 bits
So, **Logical address = 3 + 10 = 13 bits**
Physical memory has 32 frames, offset = 1024 words
Number of bits in frame# field of the Physical address = $\log_2$32 bits = 5 bits
Physical address = 5 + 10 = 15 bits
260.In a 64-bit machine, with 2 GB RAM, and 8 KB page size, how many entries will be there in the page table if it is inverted? [A]
(A) 2^{18} **(B)** 2^{20} **(C)** 2^{33} **(D)** 2^{51}
Explanation: Generally OS maintains a separate pagetable for every process and these pagetables consume a substantial amount of memory whereas an Inverted pagetable stores the paging information of all the

processes in a single page table. A simple inverted table stores one page entry per physical page(frame) and so there is a less memory overhead.
Memory size = 2 GB = 2^{31}
Page size = 8 KB = 2^{13}
Number of pagetable entries in inverted pagetable
= $2^{31}/2^{13}$ = 2^{18}
So, option (A) is correct.
261.Consider the list of page references in the time line as below:
9 6 2 3 4 4 4 4 3 4 4 2 5 8 6 8 5 5 3 2 3 3 9 6 2 7
What is the working set at the penultimate page reference if Δ is 5? [D]
(A) {8,5,3,2,9,6} **(B)** {4,3,6,2,5} **(C)** {3,9,6,2,7} **(D)** {3,9,6,2,7}

262.Suppose that the number of instructions executed between page fault is directly proportional to the number of page frames allocated to a program. If the available memory is doubled, the mean interval between page faults is also doubled. Further, consider that a normal instruction takes one microsecond, but if a page fault occurs, it takes 2001 microseconds. If a program takes 60 sec to run, during which time it gets 15,000 page faults, how long would it take to run if twice as much memory were available? [C]
(A) 60 sec **(B)** 30 sec **(C)** 45 sec **(D)** 10 sec

263.Please comment below if you find anything wrong in the above post
Consider a disk with 16384 bytes per track having a rotation time of 16 msec and average seek time of 40 msec. What is the time in msec to read a block of 1024 bytes from this disk? [B]
(A) 57 sec **(B)** 49 sec **(C)** 48 sec **(D)** 17 sec
Explanation: Transfer time(TT) = ((1024 bytes * 16 ms) / (16384 bytes)) = 1 msec
Rotational latency = 16 msec / 2 = 8 msec
seek time = 40 msec.
total time = seek time + rotational latency + transfer time
= 40 + 8 + 1 = 49 msec
264.A dynamic RAM has refresh cycle of 32 times per msec. Each refresh operation requires 100 nsec and a memory cycle requires 250 nsec. What percentage of memory's total operating time is required for refreshes? [D]
(A) 0.64 **(B)** 0.96 **(C)** 2.00 **(D)** 0.32
Explanation:
Memory cycle time = 250 nsec
memory is refreshed 32 times per msec
i.e.Number of refreshes in 1 memory cycle (i.e in 250 nsec) = $(32 * 250 * 10^{-9}) / 10^{-3} = 8 * 10^{-3}$.
Time taken for each refresh = 100 nsec
Time taken for $8 * 10^{-3}$ refreshes = $8 * 10^{-3} * 100 * 10^{-9}$.
= $8 * 10^{-10}$
Percentage of the memory cycle time used for refreshing :
= (Time taken to refresh in 1 memory cycle / Total time) * 100
= $(8 * 10^{-10} / 250 * 10^{-9}) * 100$
= 0.032 * 10
= 0.32
265.A LRU page replacement is used with four page frames and eight pages. How many page faults will occur with the reference string 0172327103 if the four frames are initially empty? [B]
(A) 6 **(B)** 7 **(C)** 8 **(D)** 5

0	1	7	2	3	2	7	1	0	3
			2	2	2	2	2	2	3
		7	7	7	7	7	7	7	7
	1	1	1	1	1	1	1	1	1
0	0	0	0	3	3	3	3	0	0
F	F	F	F	F				F	F

Explanation:

266.If the page size in a 32-bit machine is 4K bytes then the size of page table is [C]

(A) 1 M bytes **(B)** 2 M bytes **(C)** 4 M bytes **(D)** 4 K bytes

Explanation: Size of physical address = 32 bits

Size of logical address = 32 bits

page size = 4 Kbytes

Number of pages = logical address space/ size of each page

$= 2^{32} / 2^{12}$

$= 2^{20}$

page table size = number of pages * size of a page table entry

$= 2^{20} * 2^{2}$

$= 2^{22}$

267.Match the following with respect to various memory management algorithms: [D]

	List - I		List - II
(a)	Demand paging	(i)	degree of multiprogramming
(b)	Segmentation	(ii)	working set
(c)	Dynamic partitions	(iii)	supports user view of memory
(d)	Fixed partitions	(iv)	compaction

Codes :

	(a)	(b)	(c)	(d)
(1)	(iii)	(iv)	(ii)	(i)
(2)	(ii)	(iii)	(i)	(iv)
(3)	(iv)	(iii)	(ii)	(i)
(4)	(ii)	(iii)	(iv)	(i)

(A) (1) **(B)** (2) **(C)** (3) **(D)** (4)

Explanation:

- Working set is purest form of page replacement algorithm used in demand paging.
- In segmentation memory is divided into various size chunks to support user view of memory, here memory chunks are allocated to the process.
- Dynamic partition is used for compaction in which allocated objects moving together, and leaving empty space together.
- Fixed partition is dividing memory into fixed size of block for degree of multiprogramming.

So, option (D) is correct.

268.Consider a 32-bit machine where four-level paging scheme is used. If the hit ratio to TLB is 98%, and it takes 20 nanosecond to search the TLB and 100 nanoseconds to access the main memory what is effective memory access time in nanoseconds? [B]

(A) 126 **(B)** 128 **(C)** 122 **(D)** 120

Explanation: For 4 level Paging scheme, Effective Memory access Time, EAT =

$H_{TLB} * T_{TLB} + (1 - H_{TLB})[T_{TLB} + 4*T_m] + T_m]$

where, H_{TLB} = hit ratio of TLB

T_{TLB} = search time of TLB

T_m = Memory access time

Applying the formula:

EAT = (0.98 *20) + 0.02(20 + 400) + 100

EAT = 19.6 + 8.4 + 100

EAT = 128 ns

269.The character set used in Windows 2000 operating system is _____ [C]

(A) 8 bit ASCII **(B)** Extended ASCII **(C)** 16 bit UNICODE **(D)** 12 bit UNICODE

Explanation: 16 bit UNICODE character set used in Windows 2000 operating system.

Extended ASCII character encoding are 8 bit or larger encoding that include 7 bit ASCII character plus additional character.ASCII coding is 8 bit character encoding.

So, Option (C) is correct.

270.Belady's anomaly means [B]

(A) Page fault rate is constant even on increasing the number of allocated frames

(B) Page fault rate may increase on increasing the number of allocated frames

(C) Page fault rate may increase on decreasing the number of allocated frames

(D) Page fault rate may decrease on increasing the number of allocated frames

Explanation: Belady's Anomaly is situation when increasing the number of page frames allocated to a process increase the number of page faults. This generally occurs in FIFO page replacement algorithm.

271.A page fault [D]

(A) Occurs when a program accesses an available page on memory **(B)** is an error in a specific page

(C) is a reference to a page belonging to another program

(D) occurs when a program accesses a page not currently in memory

Explanation: When a page is not found in the cache, the main memory is referenced and if it is not even found in the main memory then a page fault occurs. Now, Page fault service routine is followed to bring the page to the memory form the disk and the page table is updated.

272.Dynamic address translation [C]

(A) is part of the operating system paging algorithm **(B)** is useless when swapping is used

(C) is the hardware necessary to implement paging **(D)** storage pages at a specific location on disk

Explanation: Dynamic address translation is the hardware necessary to implement paging as it converts the logical address to physical address.So, option (C) is correct.

273.Thrashing [D]

(A) always occurs on large computers **(B)** is a natural consequence of virtual memory systems

(C) can always be avoided by swapping **(D)** can be caused by poor paging algorithms

Explanation: When the degree of multi programming increases, the CPU utilization will drastically fall down and the system will spent more time only in the page replacement and the time taken to complete the execution of the process will increase. This situation in the system is called as thrashing. It can also be a consequence of poor programming algorithms.

Option (D) is correct.

274.Dirty bit for a page in a page table [A]

(A) helps avoid unnecessary writes on a paging device **(B)** helps maintain LRU information

(C) allows only read on a page **(D)** None of the above

Explanation: Dirty bit says whether the page has been modified or not. Modified means sometimes changes might have been made to the page. If a page is modified, then it has to be written back or it has to be saved back to he disk. It is set to 1 by hardware on write-access to page which is used to avoid writing when swapped out.So, option (A) is correct.

275.Checkpointing a job [A]

(A) allows it to be completed successfully **(B)** allows it to continue executing later

(C) prepares it for finishing **(D)** occurs only when there is an error in it

Explanation: Checkpointing is a method of periodically saving the state of a job so that, if for some reason, the job does not complete, it can be restarted from the saved state.So, option (B) is correct.

276.Overlaying [D]
(A) requires use of a loader **(B)** allows larger programs, but requires more effort
(C) is most used on large computers **(D)** is transparent to the user
277.A critical section is a program segment [C]
(A) which should run in a certain amount of time **(B)** which avoids deadlocks
(C) where shared resources are accessed
(D) which must be enclosed by a pair of semaphore operations P and V
Explanation: In concurrent programming, concurrent accesses to shared resources can lead to unexpected or erroneous behavior, so parts of the program where the shared resource is accessed are protected. This protected section is the critical section .So, option (C) is correct.
278.Consider a logical address space of 8 pages of 1024 words mapped into memory of 32 frames. How many bits are there in the logical address? [A]
(A) 13 bits **(B)** 15 bits **(C)** 14 bits **(D)** 12 bits
Explanation: logical address space = 8 pages of 1024 words
number of bits in logical address space = p (page bits) + d (offset bits)
number of bits = $\log_2 8 + \log_2 1024 = 3 + 10 = 13$ bits
279.The page replacement algorithm which gives the lowest page fault rate is [C]
(A) LRU **(B)** FIFO **(C)** Optimal page replacement **(D)** Second chance algorithm
Explanation: In Optimal Page replacement algorithm, pages are replaced which are not used for the longest duration of time in the future. This page replacement algorithm ensures the lowest page fault rate.Option (C) is correct.
280.What is the most appropriate function of Memory Management Unit (MMU)? [C]
(A) It is an associative memory to store TLB
(B) It is a technique of supporting multiprogramming by creating dynamic partitions
(C) It is a chip to map virtual address to physical address
(D) It is an algorithm to allocate and deallocate main memory to a process
Explanation: Memory Management Unit (MMU) is a chip to map virtual address to physical address.Option (C) s correct.
281.The term 'aging' refers to [D]
(A) booting up the priority of the process in multi-level of queue without feedback.
(B) keeping track of the following a page has been in memory for the purpose of LRU replacement
(C) letting job reside in memory for a certain amount of time so that the number of pages required can be estimated accurately.
(D) gradually increasing the priority of jobs that wait in the system for a long time to remedy infinite blocking
Explanation: In Operating systems, aging is a scheduling technique used to avoid starvation. In this, the priority of the jobs that have a longer waiting time is increased as compared to the newer processes, to avoid the starvation of older processes.Option (D) is correct.
282.The principal of the locality of reference justifies the use of [D]
(A) virtual memory **(B)** interrupts **(C)** main memory **(D)** cache memory
Explanation: Spatial Locality of reference – this says that there is chance that element will be present in the close proximity to the reference point and next time if again searched then more close proximity to the point of reference.Temporal Locality of reference – In this Least recently used algorithm will be used. Whenever there is page fault occurs within word will not only load word in main memory but complete page fault will be loaded because spatial locality of reference rule says that if you are referring any word next word will be referred in its register that's why we load complete page table so complete block will be loaded.Principle of locality of reference justifies the use of cache.Option (D) is correct.
283.Virtual memory is [B]

(A) Part of Main Memory only used for swapping
(B) A technique to allow a program, of size more than the size of main memory, to run
(C) Part of secondary storage used in program execution **(D)** None of these
Explanation: Virtual Memory is a storage allocation scheme in which secondary memory can be addressed as though it were part of main memory. The addresses a program may use to reference memory are distinguished from the addresses the memory system uses to identify physical storage sites, and program generated addresses are translated automatically to the corresponding machine addresses.
Virtual memories are incorporated to eliminate the setback of smaller RAMS with a larger process size. It allows the execution of the programs larger than the size of main memory.

284.The number of page frames that must be allocated to a running process in a virtual memory environment is determined by [A]
(A) the instruction set architecture**(B)** page size**(C)** number of processes in memory**(D)** physical memory size
285.Consider a small 2-way set-associative cache memory, consisting of four blocks. For choosing the block to be replaced, use the least recently (LRU) scheme. The number of cache misses for thefollowing sequence of block addresses is 8, 12, 0, 12, 8 [C]
(A) 2 **(B)** 3 **(C)** 4 **(D)** 5

386.The size of the ROM required to build an 8-bit adder / subtractor with mode control, carry input, carry output and two's complement overflow output is given as [B]
(A) $2^{16} \times 8$ **(B)** $2^{18} \times 10$ **(C)** $2^{16} \times 10$ **(D)** $2^{18} \times 8$
287.A specific editor has 200 K of program text, 15 K of initial stack, 50 K of initialized data, and 70 K of bootstrap code. If five editors are started simultaneously, how much physical memory is needed if shared text is used? [B]
(A) 1135 K **(B)** 335 K **(C)** 1065 K **(D)** 320 K
Explanation: Since 5 editors started simultaneously On 200k of program text along with 15 K of initial stack, 50 K of initialized data, and 70 K of bootstrap code.
Total physical memory needed = 200k + 15 K + 50 K + 70 K.
= 335 k
288.For the implementation of a paging scheme, suppose the average process size be x bytes, the page size be y bytes, and each page entry requires z bytes. The optimum page size that minimizes the total overhead due to the page table and the internal fragmentation loss is given by [C]
(A) x / 2 **(B)** xz / 2 **(C)** √(2xz) **(D)** √(xz) / 2
289.In a demand paging memory system, page table is held in registers. The time taken to service a page fault is 8 m.sec. if an empty frame is available or if the replaced page is not modified, and it takes 20 m.secs., if the replaced page is modified. What is the average access time to service a page fault assuming that the page to be replaced is modified 70 of the time? [B]
(A) 11.6 m.sec. **(B)** 16.4 m.sec. **(C)** 28 m.sec. **(D)** 14 m.sec.
Explanation: Page fault service time(PFST) = 8 msec (empty frame)
Page fault service time(PFST) = 20 msec(page is modified)
Page replacement frequency = 70% = 0.7
Page not modified = 100 – 70 = 30% = 0.3
Average access time = (0.3 * 8) + (0.7 * 20) = 2.4 + 14 = 16.4 m.sec
290.Which methods are utilized to control the access to an object in multi-threaded programming? [B]
(A) Asynchronized methods **(B)** Synchronized methods

(C) Serialized methods **(D)** None of the above

291.Processes P1 and P2 have a producer-consumer relationship, communicating by the use of a set of shared buffers.

P1: repeat

Obtain an empty buffer

Fill it

Return a full buffer

forever

P2: repeat

Obtain a full buffer

Empty it

Return an empty buffer

forever

Increasing the number of buffers is likely to do which of the following? [C]

I. Increase the rate at which requests are satisfied (throughput)
II. Decrease the likelihood of deadlock
III. Increase the ease of achieving a correct implementation

(A) III only **(B)** II only **(C)** I only **(D)** II and III only

292.Please comment below if you find anything wrong in the above post

Determine the number of page faults when references to pages occur in order – 1, 2, 4, 5, 2, 1, 2, 4. Assume that the main memory can accommodate 3 pages and the main memory already has the pages 1 and 2, with page 1 having brought earlier than page 2.(assume LRU algorithm is used) [B]

(A) 3 **(B)** 4 **(C)** 5 **(D)** None of these

Explanation: The references to pages occur in order – 1, 2, 4, 5, 2, 1, 2, 4 can be seen in the figure:

1	2	4	5	2	1	2	4
2	1	1	2	4	5	5	1
1	2	2	4	5	2	1	2
		4	5	2	1	2	4
X	**X**	**P.F**	**P.F**	**X**	**P.F**	**X**	**P.F**

293. In a paged memory, the page hit ratio is 0.40. The time required to access a page in secondary memory is equal to 120 ns. The time required to access a page in primary memory is 15 ns. The average time required to access a page is _____. [D]

(A) 105 **(B)** 68 **(C)** 75 **(D)** 78

Explanation: Average access time = hit ratio * primary memory access time + (1 – hit ratio) * secondary memory access time

Average access time = 0.4 * 15 + 0.6 * 120

Average access time = 6 + 72

Average access time = 78.

294.Which of the following statements are true ? [C]

(a) External Fragmentation exists when there is enough total memory space to satisfy a request but the available space is contiguous.

(b) Memory Fragmentation can be internal as well as external.
(c) One solution to external Fragmentation is compaction.
Code:

(A) (a) and (b) only **(B)** (a) and (c) only **(C)** (b) and (c) only **(D)** (a), (b) and (c)

Explanation: External Fragmentation exists when there is enough total memory space to satisfy a request but the available space is contiguous. This is a false statement because external fragmentation occurs due to non contigious available space.Memory Fragmentation can be internal as well as external.**True.** One solution to external Fragmentation is compaction or shuffle memory contents **True.**So, option (C) is correct.

295.Page information in memory is also called as Page Table. The essential contents in each entry of a page table is/are _____. [C]

(A) Page Access information **(B)** Virtual Page number
(C) Page Frame number **(D)** Both virtual page number and Page Frame Number

Explanation: Page information in memory is also called as Page Table. The essential contents in each entry of a page table is page frame number.

296.Consider a virtual page reference string 1, 2, 3, 2, 4, 2, 5, 2, 3, 4. Suppose LRU page replacement algorithm is implemented with 3 page frames in main memory. Then the number of page faults are_____. [B]

(A) 5 **(B)** 7 **(C)** 9 **(D)** 10

Explanation: Page replacement diagram for LRU:

Deadlock

297.Suppose n processes, P1, …. Pn share m identical resource units, which can be reserved and released one at a time. The maximum resource requirement of process Pi is Si, where Si > 0. Which one of the following is a sufficient condition for ensuring that deadlock does not occur? [C]

(a) $\forall i, s_i < m$ (b) $\forall i, s_i < n$

(c) $\sum_{i=1}^{n} s_i < (m+n)$ (d) $\sum_{i=1}^{n} s_i < (m * n)$

(A) A **(B)** B **(C)** C **(D)** D

298.Consider the following snapshot of a system running n processes. Process i is holding Xi instances of a resource R, 1 <= i <= n. currently, all instances of R are occupied. Further, for all i, process i has placed a request for an additional Yi instances while holding the Xi instances it already has. There are exactly two processes p and q such that Yp = Yq = 0. Which one of the following can serve as a necessary condition to guarantee that the system is not approaching a deadlock? [B]

(A) min (Xp, Xq) < max (Yk) where k != p and k != q **(B)** Xp + Xq >= min (Yk) where k != p and k != q
(C) max (Xp, Xq) > 1 **(D)** min (Xp, Xq) > 1

Explanation: Since both p and q don't need additional resources, they both can finish and release Xp + Xq resources without asking for any additional resource. If the resources released by p and q are sufficient for another process waiting for Yk resources, then system is not approaching deadlock.

299.Three concurrent processes X, Y, and Z execute three different code segments that access and update certain shared variables. Process X executes the P operation (i.e., wait) on semaphores a, b and c; process Y executes the P operation on semaphores b, c and d; process Z executes the P operation on semaphores c, d, and a before entering the respective code segments. After completing the execution of its code segment, each process invokes the V operation (i.e., signal) on its three semaphores. All semaphores are binary semaphores initialized to one. Which one of the following represents a deadlockfree order of invoking the P operations by the processes? (GATE CS 2013) [B]

(A) X: P(a)P(b)P(c) Y:P(b)P(c)P(d) Z:P(c)P(d)P(a) **(B)** X: P(b)P(a)P(c) Y:P(b)P(c)P(d) Z:P(a)P(c)P(d)
(C) X: P(b)P(a)P(c) Y:P(c)P(b)P(d) Z:P(a)P(c)P(d) **(D)** X: P(a)P(b)P(c) Y:P(c)P(b)P(d) Z:P(c)P(d)P(a)

Explanation: Option A can cause deadlock. Imagine a situation process X has acquired a, process Y has acquired b and process Z has acquired c and d. There is circular wait now.
Option C can also cause deadlock. Imagine a situation process X has acquired b, process Y has acquired c and process Z has acquired a. There is circular wait now.Option D can also cause deadlock. Imagine a situation process X has acquired a and b, process Y has acquired c. X and Y circularly waiting for each other.

300.Consider option A) for example here all 3 processes are concurrent so X will get semaphore a, Y .A system has n resources R_0,…,R_{n-1},and k processes P_0,….P_{k-1}.The implementation of the resource request logic of each process P_i is as follows:

```
if (i % 2 == 0) {
    if (i < n) request Ri
    if (i+2 < n) request Ri+2
}
else {
    if (i < n) request Rn-i
    if (i+2 < n) request Rn-i-2
}
```

In which one of the following situations is a deadlock possible? [B]
(A) n=40, k=26 **(B)** n=21, k=12 **(C)** n=20, k=10 **(D)** n=41, k=19
Explanation:
Option B is answer

No. of resources, n = 21

No. of processes, k = 12

Processes {P0, P1....P11} make the following Resource requests:

{R0, R20, R2, R18, R4, R16, R6, R14, R8, R12, R10, R10}

For example P0 will request R0 (0%2 is = 0 and 0< n=21).

Similarly, P10 will request R10.

P11 will request R10 as n - i = 21 - 11 = 10.

As different processes are requesting the same resource, deadlockmay occur.

301.Consider a system with 4 types of resources R1 (3 units), R2 (2 units), R3 (3 units), R4 (2 units). A non-preemptive resource allocation policy is used. At any given instance, a request is not entertained if it cannot be completely satisfied. Three processes P1, P2, P3 request the sources as follows if executed independently.

Process P1:
t=0: requests 2 units of R2 t=1: requests 1 unit of R3 t=3: requests 2 units of R1
t=5: releases 1 unit of R2 and 1 unit of R1. t=7: releases 1 unit of R3
t=8: requests 2 units of R4 t=10: Finishes

Process P2:
t=0: requests 2 units of R3 t=2: requests 1 unit of R4 t=4: requests 1 unit of R1
t=6: releases 1 unit of R3 t=8: Finishes

Process P3:
t=0: requests 1 unit of R4 t=2: requests 2 units of R1 t=5: releases 2 units of R1
t=7: requests 1 unit of R2 t=8: requests 1 unit of R3 t=9: Finishes

Which one of the following statements is TRUE if all three processes run concurrently starting at time t=0? **[A]**
(A) All processes will finish without any deadlock **(B)** Only P1 and P2 will be in deadlock.
(C) Only P1 and P3 will be in a deadlock. **(D)** All three processes will be in deadlock

Explanation:
We can apply the following Deadlock Detection algorithm and see that there is no process waiting indefinitely for a resource.

303.Which of the following is NOT true of deadlock prevention and deadlock avoidance schemes? [A]
(A) In deadlock prevention, the request for resources is always granted if the resulting state is safe
(B) In deadlock avoidance, the request for resources is always granted if the result state is safe
(C) Deadlock avoidance is less restrictive than deadlock prevention
(D) Deadlock avoidance requires knowledge of resource requirements a priori

Explanation:
Deadlock Prevention: Deadlocks can be prevented by preventing at least one of the four required conditions:

1. **Mutual Exclusion** – not required for sharable resources; must hold for non-sharable resources.
2. **Hold and Wait** – must guarantee that whenever a process requests a resource, it does not hold any other resources. Require process to request and be allocated all its sources before it begins execution, or allow process to request resources only when the process has none. Low resource utilization; starvation possible. Restrain the ways request can be made.
3. **No Pre-emption** – If a process that is holding some resources requests another resource that cannot be immediately allocated to it, and then all resources currently being held are released. Pre-empted resources are added to the list of resources for which the process is waiting. Process will be restarted only when it can regain its old resources, as well as the new ones that it is requesting.
4. **Circular Wait** – impose a total ordering of all resource types, and require that each process requests resources in an increasing order of enumeration.

Deadlock Avoidance:
When a scheduler sees that starting a process or granting resource requests may lead to future deadlocks, then that process is just not started or the request is not granted. The deadlock-avoidance algorithm dynamically examines the resource-allocation state to ensure that there can never be a circular-wait condition. Resource-allocation state is defined by the number of available and allocated resources, and the maximum demands of the processes.

Choice of the question:
(A) In deadlock prevention, the request for resources is always granted if the resulting state is safe. false, Deadlock prevention scheme handles deadlock by making sure that one of the four necessary conditions don't occur. In deadlock prevention, the request for a resource may not be granted even if the resulting state is safe.

(B) In deadlock avoidance, the request for resources is always granted if the result state is safe. true, As in Deadlock avoidance, if resultant state is safe than request for resource is granted as being in a safe state, it can hold other resources now.

(C) Deadlock avoidance is less restrictive than deadlock prevention. true, As in Deadlock prevention, request for a resource may not be granted even if the resulting state is safe. but in deadlock avoidance, request for a resource is granted if the resulting state is safe.

(D) Deadlock avoidance requires knowledge of resource requirements a priori true, deadlock avoidance checks any chance of deadlock means even if the system is in safe state, it checks that after allocating requested resource, the system is not in deadlocked state. So deadlock avoidance requires knowledge of resource requirements a priori.

304.A system contains three programs and each requires three tape units for its operation. The minimum number of tape units which the system must have such that deadlocks never arise is _____. [B]

(A) 6 **(B)** 7 **(C)** 8 **(D)** 9

Explanation: If 6 resources are there then it may be possible that all three process have 2 resources and waiting for 1 more resource. Therefore, all of them will have to wait indefinitely. If 7 resources are there, then atleast one must have 3 resources so deadlock can never occur.

305.Consider the following snapshot of a system running n processes. Process i is holding Xi instances of a resource R, 1 <= i <= n. currently, all instances of R are occupied. Further, for all i, process i has placed a request for an additional Yi instances while holding the Xi instances it already has. There are exactly two processes p and q such that Yp = Yq = 0. Which one of the following can serve as a necessary condition to guarantee that the system is not approaching a deadlock? [B]

(A) min (Xp, Xq) < max (Yk) where k != p and k != q **(B)** Xp + Xq >= min (Yk) where k != p and k != q

(C) max (Xp, Xq) > 1 **(D)** min (Xp, Xq) > 1

Explanation:

Deadlock refers to a specific condition when two or more processes are each waiting for another to release a resource, or more than two processes are waiting for resources.

Solution:

Necessary condition to guarantee no deadlock which means without satisfying this condition, no deadlock is possible. Both the process p and q have no additional requirement; they both can be finished releasing Xp + Xq resources without asking for any additional resource.

Using this, we can finish one more process only if condition B is satisfied. If the resources released by p and q are sufficient for another process waiting for Yk resources, then system is not approaching deadlock.

i.e Xp+Xq > min (Yk) where k != p and k != q

Note: Option B just ensured that the system can proceed from the current state. It does not guarantee that there won't be a deadlock before all processes are finished.

307.Suppose n processes, P1, …. Pn share m identical resource units, which can be reserved and released one at a 306.time. The maximum resource requirement of process Pi is Si, where Si > 0. Which one of the following is a sufficient condition for ensuring that deadlock does not occur? [C]

(a) $\forall i, s_i < m$ (b) $\forall i, s_i < n$

(c) $\sum_{i=1}^{n} s_i < (m+n)$ (d) $\sum_{i=1}^{n} s_i < (m*n)$

(A) A **(B)** B **(C)** C **(D)** D

307.Let m[0]…m[4] be mutexes (binary semaphores) and P[0] …. P[4] be processes.

Suppose each process P[i] executes the following:

```
wait (m[i]); wait(m[(i+1) mode 4]);

------

release (m[i]); release (m[(i+1)mod 4]);
```

This could cause: [B]

(A) Thrashing **(B)** Deadlock **(C)** Starvation, but not deadlock **(D)** None of the above

308.Which of the following is NOT a valid deadlock prevention scheme? (GATE CS 2000)[C]

(A) Release all resources before requesting a new resource

(B) Number the resources uniquely and never request a lower numbered resource than the last one requested.

(C) Never request a resource after releasing any resource

(D) Request and all required resources be allocated before execution.

309.A system has 6 identical resources and N processes competing for them. Each process can request atmost 2 resources. Which one of the following values of N could lead to a deadlock? [D]

(A) 1 **(B)** 2 **(C)** 3 **(D)** 4

310.Below seems to be the worst case situation. In below situation P1 and P2 can continue as they have 2 resources each

P1 --> 2 P2 --> 2 P3 --> 1 P4 --> 1

Consider the following policies for preventing deadlock in a system with mutually exclusive resources.

I. Processes should acquire all their resources at the beginning of execution. If any resource is not available, all resources acquired so far are released.

II. The resources are numbered uniquely, and processes are allowed to request for resources only in increasing resource numbers.

III. The resources are numbered uniquely, and processes areallowed to request for resources only in decreasing resource numbers.

IV. The resources are numbered uniquely. A process is allowedto request only for a resource with resource number larger than its currently held resources.

Which of the above policies can be used for preventing deadlock? [D]

(A) Any one of I and III but not II or IV **(B)** Any one of I, III and IV but not II

(C) Any one of II and III but not I or IV **(D)** Any one of I, II, III and IV

Explanation: If Ist is followed, then hold and wait will never happen.

II, III and IV are similar. If any of these is followed, cyclic wait will not be possible.

311.Two shared resources R_1 and R_2 are used by processes P_1 and P_2. Each process has a certain priority for accessing each resource. Let T_{ij} denote the priority of P_i for accessing R_j. A process P_i can snatch a resource R_h from process P_j if T_{ik} is greater than T_{jk}.

Given the following :

1. $T_{11} > T_{21}$ 2.$T_{12} > T_{22}$ 3.$T_{11} < T_{21}$ 4.$T_{12} < T_{22}$

Which of the following conditions ensures that P_1 and P_2 can never deadlock? [C]

(A) (I) and (IV) **(B)** (II) and (III) **(C)** (I) and (II) **(D)** None of the above

Explanation:

If all the resources are allocated to one process then deadlock will never occur.

So, if we allocate both R1 and R2 to process P1 or both R1 and R2 to process P2 then deadlock can be prevented.When one process completes its execution then both the resources are allocated to the other process.So, either condition (I) and (II) or condition (III) and (IV) ensures that deadlock never occurs.

313.Consider the following proposed solution for the critical section problem. There are n processes: P0 …Pn−1. In the code, function pmax returns an integer not smaller than any of its arguments. For all i, t[i] is initialized to zero.

```
do {
      c[i]=1; t[i] = pmax(t[0],...,t[n-1])+1; c[i]=0;
      for every j ≠ i in {0,...,n-1} {
          while (c[j]);
          while (t[j] != 0 && t[j]<=t[i]);
      }
      Critical Section;
      t[i]=0;
      Remainder Section;
} while (true);
```

Which one of the following is TRUE about the above solution? [A]

(A) At most one process can be in the critical section at any time **(B)** The bounded wait condition is satisfied

(C) The progress condition is satisfied **(D)** It cannot cause a deadlock

Explanation:

Mutual exclusion is satisfied:

All other processes j started before i must have value (i.e. t[j]) less than the value of process i (i.e. t[i]) as function pMax() return a integer not smaller than any of its arguments. So if anyone out of the processes j have positive value will be executing in its critical section as long as the condition t[j] > 0 && t[j] <= t[i] within

while will persist. And when this j process comes out of its critical section, it sets t[j] = 0; and next process will be selected in for loop.So when i process reaches to its critical section none of the processes j which started earlier before process i is in its critical section. This ensure that only one process is executing its critical section at a time.

Deadlock and progress are not satisfied: while (t[j] != 0 && t[j] <=t[i]); because of this condition deadlock is possible when value of j process becomes equals to the value of process i (i.e t[j] == t[i]). because of the deadlock progess is also not possible

(i.e. Progess == no deadlock) as no one process is able to make progress by stoping other process.

Bounded waiting is also not satisfied: In this case both deadlock and bounded waiting to be arising from the same reason as if t[j] == t[i] is possible then starvation is possible means

infinite waiting.

314.Processes P_1 and P_2 use critical_flag in the following routine to achieve mutual exclusion. Assume that critical_flag is initialized to FALSE in the main program.

```
get_exclusive_access ( )
{
if (critical _flag == FALSE) {
critical_flag = TRUE ;
critical_region () ;
critical_flag = FALSE;
}
}
```

Consider the following statements.

i. It is possible for both P_1 and P_2 to access critical_region concurrently.

ii. This may lead to a deadlock.

Which of the following holds? [C]

(A) (i) is false and (ii) is true **(B)** Both (i) and (ii) are false **(C)** (i) is true and (ii) is false **(D)** Both (i) and (ii) are true

```
get_exclusive_access()
{
    if(critical_flage == False){
      critical_flage == True;
      critical_region ( );
      critical_fflage == False;
    }
}
```

Explanation:

Say P_1 starts first and executes statement 1, after that system context switches to P_2 (before executing statement 2), and it enters inside if statement, since the flag is still false.

So now both processes are in critical section!! so (i) is true.. (ii) is false

By no way it happens that flag is true and no process' are inside the if clause, if someone enters the critical section, it will definitely make flag = false. So no deadlock.

315.A counting semaphore was initialized to 10. Then 6 P (wait) operations and 4 V (signal) operations were 315.completed on this semaphore. The resulting value of the semaphore is [B]

(A) 0 **(B)** 8 **(C)** 10 **(D)** 12

Explanation: Initially we have semaphore value = 10

Now we have to perform 6 p operation means when we perform one p operation it decreases the semaphore values to one.So after performing 6 p operation we get, semaphore values = 10 – 6 = 4 and now we have to perform 4 v operation means when we perform one V operation it increases the semaphore values to one. So

after performing 4 v operation we get, semaphore values = 4 + 4 = 8.Option (B) is correct.

316.A computer has six tape drives, with n processes competing for them. Each process may need two drives. What is the maximum value of n for the system to be deadlock free? [B]

(A) 6 **(B)** 5 **(C)** 4 **(D)** 3

Explanation: Given tape drive = 6 and each process may need 2 drive.

When we give 1 drive to 1 process then total process will be 6 but in this case there will definitely deadlock occur because every process contain 1 drive and waiting for another drive which is hold by other process therefore when we reduce 1 process then system to be deadlock free.Hence maximum value of n = 6 – 1 = 5.Option (B) is correct.

317.An operating system contains 3 user processes each requiring 2 units of resource *R*. The minimum number of units of *R* such that no deadlocks will ever arise is [C]

(A) 3 **(B)** 5 **(C)** 4 **(D)** 6

Explanation: Total process = 3 and each require 2 units of resource.

If we give 1 resource to 1 process then total resource = 1 + 1 + 1 = 3 but in this case deadlock will definitely occur because each process hold 1 unit resource and waiting for another resource so if we increase 1 more resource (3+1=4) then deadlocks will ever arise (i.e., when process 1 complete their execution then it free 2 resource and this 2 resource will used by another process.).Option (B) is correct.

318.Each Process P_i, i= 1…….9 is coded as follows

```
repeat
   P(mutex)
   {Critical section}
   V(mutex)
forever
```

The code for P_{10} is identical except it uses V(mutex) in place of P(mutex). What is the largest number of processes that can be inside the critical section at any moment? [D]

(A) 1 **(B)** 2 **(C)** 3 **(D)** None of above

Explanation: Consider code for i= 1 to 9

Initially mutex value set to 1, so it can allow only 1 process at a time. Now p1 enter into critical section and by this time remaining all are in block state. (i.e blocked processes= 2,3,4,5,6,7,8,9). But consider code for 10th process, it tells that unblock any process because it contain operation. Because of this it can unblock the processes and send it to the critical section. By doing this all processes can enter into critical section. So, finally there are 10 processes at maximum can enter into critical section.

Alternative Way-

There is loop which runs forever So P10 runs forever. And it has capability to put all other processes to Critical Section. Moreover they Asked Maximum Number of process that can be Present at critical Section So Answer is 10.Option (D) is correct.

319.An operating system handles requests to resources as follows.

A process (which asks for some resources, uses them for some time and then exits the system) is assigned a unique timestamp are when it starts. The timestamps are monotonically increasing with time. Let us denote the timestamp of a process P by TS(P).

When a process P requests for a resource the OS does the following:

(i) If no other process is currently holding the resource, the OS awards the resource to P.

(ii) If some process Q with *TS(Q)<TS(P)* is holding the resource, the OS makes P wait for the resources.

(iii) If some process Q with *TS(Q)>TS(P)* is holding the resource, the OS restarts Q and awards the resources to P.

(Restarting means taking back the resources held by a process, killing it and starting it

again with the same timestamp)

When a process releases a resource, the process with the smallest timestamp (if any) amongst those waiting for the resource is awarded the resource.

a). Can a deadlock ever arise? If yes, show how. If not, prove it.

b). Can a process P ever starve? If yes, show how. If not, prove it.

320.A critical section is a program segment [C]

(A) which should run in a certain specified amount of time **(B)** which avoids deadlocks

(C) where shared resources are accessed

(D) which must be enclosed by a pair of semaphore operations, P and V

Explanation: Critical section is the part of program where shared resources are accessed by process.It also contain shared variable or shared data and rest portion of the program is called non-critical section or remaining section.Option (C) is correct.

321.A solution to the Dining Philosophers Problem which avoids deadlock is: [C]

(A) ensure that all philosophers pick up the left fork before the right fork

(B) ensure that all philosophers pick up the right fork before the left fork

(C) ensure that one particular philosopher picks up the left fork before the right fork, and that all other philosophers pick up the right fork before the left fork **(D)** None of the above

Explanation: In Dining Philosophers Problem it ensure that one particular philosopher picks up the left fork before the right fork, and that all other philosophers pick up the right fork before the left fork which avoids the deadlock condition.Option (C) is correct.

322.A computer system uses the Banker's Algorithm to deal with deadlocks. Its current state is shown in the table below, where P0, P1, P2 are processes, and R0, R1, R2 are resources types.

Maximum Need				Current Allocation				Available		
	R0	R1	R2		R0	R1	R2	R0	R1	R2
P0	4	1	2	P0	1	0	2	2	2	0
P1	1	5	1	P1	0	3	1			
P2	1	2	3	P2	1	0	2			

a) Show that the system can be in this state.

b) What will the system do on a request by process P0 for one unit of resource type R1?

323.A system shares 9 tape drives. The current allocation and maximum requirement of tape drives for 4 processes are shown below:

Process	Maximum need	Current allocation
P1	9	3
P2	6	1
P3	5	3
P4	10	0

Which of the following best describes the current state of the system? [C]

(A) Safe, Deadlocked **(B)** Safe, Not Deadlocked

(C) Not Safe, Deadlocked **(D)** Not Safe, Not Deadlocked

Explanation: There are 9 tapes and allocated (3+1+3+0 =) 7, so remaining available tapes are 9 – 7 = 2.

Process	Maximum need	Current allocation	Current available	Remaining need
P1	9	3	2	6
P2	6	1		5
P3	5	3		2
P4	10	0		10

Now we can satisfies only process P3 first which require only 2 tapes, then current available will be total 5.Similarly, we can satisfies only process P2 which require only 5 tapes, then current available will be total 6.Then, we can satisfies only process P1 which require only 6 tapes, then current available will be total 9.But we can not satisfied remaining need of process P4, because it requires 10 tapes and remaining need 9.

324.Consider a system with 3 processes that share 4 instances of the same resource type. Each process can request a maximum of K instances. Resource instances can be requested and released only one at a time. The largest value of K that will always avoid deadlock is _____ . [B]

Note – This was Numerical Type question.

(A) 1 **(B)** 2 **(C)** 3 **(D)** 4

Explanation:

Given,

Number of processes (P) = 3

Number of resources (R) = 4.Since deadlock-free condition is:

$R \geq P(N - 1) + 1$

Where R is total number of resources,

P is the number of processes, and

N is the max need for each resource.

$4 \geq 3(N - 1) + 1$

$3 \geq 3(N - 1)$

$1 \geq (N - 1)$

$N \leq 2$

Therefore, the largest value of K that will always avoid deadlock is **2**.Option (B) is correct.

325.Consider the following solution to the producer-consumer synchronization problem. The shared buffer size is N. Three semaphores *empty, full* and *mutex* are defined with respective initial values of 0, N and 1. Semaphore *empty* denotes the number of available slots in the buffer, for the consumer to read from. Semaphore *full* denotes the number of available slots in the buffer, for the producer to write to. The placeholder variables, denoted by P, Q, R and S, in the code below can be assigned either *empty* or *full*. The valid semaphore operations are: *wait()* and *sigmal()*.

Producer:	Consumer:

do{
wait(P);
wait(mutex);
//Add item to buffer
signal(mutex);
signal(Q);
}while(1);
	do{
	wait(R);
	wait(mutex);
	//Consume item from buffer
	signal(mutex);
	signal(S);
	}while(1);

326.Which one of the following assignments to P, Q, R and S will yield the correct solution? [C]

(A) P: *full*, Q: *full*, R: *empty*, S: *empty* **(B)** P: *empty*, Q: *empty*, R: *full*, S: *full*

(C) P: *full*, Q: *empty*, R: *empty*, S: *full* **(D)** P: *empty*, Q: *full*, R: *full*, S: *empty*

Explanation: Given,

Empty = 0

Full = N

Mutex = 1

Since value of *empty* semaphore is 0, so you can not wait *empty* semaphore in first attempt.

Note – *empty* semaphore denotes the number of filled slots, so producer process must deal with *empty* and mutex semaphores. *full* semaphore denotes the number of empty slots so consumer process must deal with *full* and mutex semaphores.

Option (A) causes starvation.
OPtion (B) causes starvation.
Option (D) since number of filled slots are 0 initially denoted by *empty* semaphore, so consumer process can not consume. So, this implimentation is wrong.
Only P: *empty*, Q: *full*, R: *full*, S: *empty* is correct order to ensure deadlock-free and starvation free implementation.Option (C) is correct.
327.In a system, there are three types of resources: E, F and G. Four processes P_0, P_1, P_2 and P_3 execute concurrently. At the outset, the processes have declared their maximum resource requirements using a matrix named Max as given below. For example, Max[P_2, F] is the maximum number of instances of F that P_2 would require. The number of instances of the resources allocated to the various processes at any given state is given by a matrix named Allocation.
Consider a state of the system with the Allocation matrix as shown below, and in which 3 instances of E and 3 instances of F are the only resources available.

Allocation			
	E	*F*	*G*
P_0	1	0	1
P_1	1	1	2
P_2	1	0	3
P_3	2	0	0

Max			
	E	*F*	*G*
P_0	4	3	1
P_1	2	1	4
P_2	1	3	3
P_3	5	4	1

From the perspective of deadlock avoidance, which one of the following is true? [A]

(A) The system is in *safe* state
(B) The system is not in *safe* state, but would be *safe* if one more instance of E were available
(C) The system is not in *safe* state, but would be *safe* if one more instance of F were available
(D) The system is not in *safe* state, but would be *safe* if one more instance of G were available

Explanation:

	MAX			Allocation			Need			Available		
	E	*F*	*G*	*E*	*F*	*G*	*E*	*F*	*G*	*E*	*F*	*G*
P0	4	3	1	1	0	1	3	3	0	3	3	0
P1	2	1	4	1	1	2	1	0	2			
P2	1	3	3	1	0	3	0	3	0			
P3	5	4	1	2	0	0	3	4	1			

Available (3, 3, 0), which can satisfy either P0 or P2.
Take P0 <3, 3, 0>. After completion we have (3, 3, 0) + (1, 0, 1) = (4, 3, 1)
Take P2 <0, 3, 0>. After completion we have (4, 3, 1) + (1, 0, 3) = (5, 3, 4)
Take P1 <1, 0, 2>. After completion we have (5, 3, 4) + (1, 1, 2) = (6, 4, 6)
Take P3 <3, 4, 1>. After completion we have (6, 4, 6) + (2, 0, 0) = (8, 4, 6)

Safe sequence : **P0–>P2–>P1–>P3**
328.Which of the following is not true with respect to deadlock prevention and deadlock avoidance schemes ? [A]

(A) In deadlock prevention, the request for resources is always granted if resulting state is safe
(B) In deadlock avoidance, the request for resources is always granted, if the resulting state is safe
(C) Deadlock avoidance requires knowledge of resource requirements a priori
(D) Deadlock prevention is more restrictive than deadlock avoidance

Explanation: In Deadlock Prevention mechanism a set of methods are devised to ensure at least one of the necessary conditions “Mutual Exclusion”, “Hold and Wait”, “Circular Wait”, “Non-preemption” can not

hold.Option (A) is correct.

329.Which of the following is not true with respect to deadlock prevention and deadlock avoidance schemes? [A]

(A) In deadlock prevention, the request for resources is always granted if resulting state is safe
(B) In deadlock avoidance, the request for resources is always granted, if the resulting state is safe
(C) Deadlock avoidance requires knowledge of resource requirements a priori
(D) Deadlock prevention is more restrictive than deadlock avoidance

330.What problem is solved by Dijkstra banker's algorithm? [D]

(A) Cache coherence **(B)** Mutual exclusion **(C)** Deadlock recovery **(D)** Deadlock avoidance

331.With single resource, deadlock occurs [D]

(A) if there are more than two processes competing for that resources
(B) if there are only two processes competing for that resources
(C) if there is a single process competing for that resources **(D)** none of these

Explanation: There are 4 conditions required for a deadlock to occur:
1) Mutual Exclusion 2) No pre-emption 3) Hold and wait 4) Circular wait

331.When there is only one resource, the conditions of hold and wait and circular wait get eliminated. With the assumption that no process can use a resource for infinite amount of time, whenever a process will finish its execution , another process can get the resource. So a deadlock can't occur.So, option (D) is correct.

332.What is the minimum number of resources required to ensure that deadlock will never occur, if there are currently three processes P1, P2 and P3 running in a system whose maximum demand for the resources of same type are 3, 4, and 5 respectively. [D]

(A) 3 **(B)** 7 **(C)** 9 **(D)** 10

Explanation: let the resources needed by P1 = R1 =3,
the number of resources needed by P2 = R2 = 4 and so on..
Minimum resources required to ensure that deadlock will never occur = (R1-1) + (R2 1) + (R3-1) + 1= (3-1) + (4-1)+ (5-1) + 1 = 10.Option (D) is correct.

333.Which of the following is not a necessary condition for deadlock? [B]

(A) Mutual exclusion **(B)** Reentrancy **(C)** Hold and wait **(D)** No pre-emption

Explanation: There are 4 necessary conditions for Deadlock to occur:
1) Mutual Exclusion 2) No pre-emption 3) Hold and Wait 4) Circular wait

334.Consider the following process and resource requirement of each process.

	Type 1		Type 2	
Process	Used	Max	Used	Max
P1	1	2	1	3
P2	1	3	1	2
P3	2	4	1	4

Predict the state of this system, assuming that there are a total of 5 instances of resource type 1 and 4 instances of resource type 2. [C]

(A) Can go to safe or unsafe state based on sequence **(B)** Safe state **(C)** Unsafe state **(D)** Deadlock state

Explanation: Need Matrix of the processes:

	Type 1			Type 2		
Process	Used	Max	Need	Used	Max	Need
P1	1	2	1	1	3	2

P2	1	3	2	1	2	2
P3	2	4	2	1	4	3

Available resources : type1 = 5 – 4 = 1 and type2 = 4 – 3 = 1

But with one resource of type1 and type2 each, none of the process request can be fulfilled, so it is an unsafe state.Option (C) is correct.

335.A total of 9 units of a resource type available, and given the safe state shown below, which of the following sequence will be a safe state? [D]

Process	Used	Max
P1	2	7
P2	1	6
P3	2	5
P4	1	4

(A) (P4, P1, P3, P2) **(B)** (P4, P2, P1, P3) **(C)** (P4, P2, P3, P1) **(D)** (P3, P1, P2, P4)

Explanation: Applying Banker's Algorithm, Need matrix of the processes are:

Process	Used	Max	Need
P1	2	7	5
P2	1	6	5
P3	2	5	3
P4	1	4	3

Currently Available resources = Available – Allocated resources = 9 – 6 = 3

If the request of P4 is granted first then it would release a maximum of 4 resources after its execution, and if P1 or P2 are allocated next then their requests can not be fulfilled as both of them require 5 resources each.So, that eliminates Options (A), (B) and (C).Option (D) is correct.

336.Consider a system having "n" resources of same type. These resources are shared by 3 processes, A, B, C. These have peak demands of 3, 4, and 6 respectively. For what value of "n" deadlock won't occur [D]

(A) 15 **(B)** 9 **(C)** 10 **(D)** 11

Explanation: The value of n for which the deadlock can't occur = n(k-1) + 1

Number of min resources required = (3-1) + (4-1) + (6-1) + 1 = 11

337.When a process is rolled back as a result of deadlock the difficulty which arises is [A]

(A) Starvation **(B)** System throughput **(C)** Low device utilization **(D)** Cycle stealing

Explanation: When a process is rolled back as a result of deadlock, the difficulty which arises is starvation because when we rollback the process then there may be possibility that again deadlock occur after that again process will wait for indefinite amount of time that's why it goes to starvation.On the other hand Low device utilization and cycle stealing is not a related concept to the rollback of process as a result of deadlock.

338.In which of the following four necessary conditions for deadlock processes claim exclusive control of the resources they require? [B]

(A) no preemption **(B)** mutual exclusion **(C)** circular wait **(D)** hold and wait

Explanation: Mutual Exclusion is a condition when one or more than one resource are non-sharable (Only one process can use at a time) i.e. processes claim exclusive control of the resources they require.So, option (B) is correct.

339.Consider a system having 'm' resources of the same type. The resources are shared by 3 processes A, B, C, which have peak time demands of 3, 4, 6 respectively. The minimum value of 'm' that ensures that deadlock will never occur is [A]

(A) 11 **(B)** 12 **(C)** 13 **(D)** 14

Explanation: Minimum resources required to avoid deadlock = (m1 – 1) + (m2 – 1) +..+ (my – 1) + 1
where m = resource required by process
y = number of processes
so, Number of resources that ensures that deadlock will never occur is = (3-1) + (4-1) + (6-1) + 1 = 11

340.Consider a system having m resources of the same type. These resources are shared by 3 processes A, B, C which have peak time demands of 3, 4, 6 respectively. The minimum value of m that ensures deadlock will never occur is [A]

(A) 11 **(B)** 12 **(C)** 13 **(D)** 14

Explanation: Let the resources needed by P1 = R1 = 3, the number of resources needed by P2 = R2 = 4 and so on.
Minimum resources required to ensure that deadlock will never occur = (R1-1) + (R2-1) + (R3-1) + 1 = (3-1) + (4-1) + (6-1) + 1 = 11