1. Identify whether the following statements are TRUE or FALSE. If the statement is FALSE, correct it and justify the corrected sentence. If the statement is TRUE, justify it. Restrict the justification to a few (less than five) sentences. [10*1=10]

1.1 As compared to CPU-bound process, OS gives higher priority for I/O bound process.

True.

because. if I/O bound process gets executed first in cpu ~~on~~ then it will be immediately ~~went~~ go to waiting state so that now cpu bound process can run on cpu. Now both of them run simultaneously ~~on~~ one in cpu, one in I/O.

1.2 With monitors, it is possible to reduce busy waiting over semaphores.

→ True. ✗

because mutual exclusion it automatically handled by monitors.

0

1.3 It is easier to port micro-kernel operating system from one hardware to another hardware as compared to non-micro-kernel operating system.

→ ~~True~~. False. ✗

because all. the functional of O.S are micro kernel. O.S.

1.4 The notion of "load balancing" counteracts with the notion of "processor affinity".

true.

load balancing:- equal load on all the cores.

1 process affinity :- a particular process want to run on a particular core.

1.5 Global frame allocation algorithm gives better throughput over local frame allocation algorithm.

0.5 true

1.6 From the system side, it is easy to provide Session Semantics feature as compared to UNIX semantics feature.

true. X

0

1.7 Fourier theorem helps to improve the efficiency of the channel/link.

true.

because the signal could be written as addition of various signals.

signal. = 3 + a sin tn + b cos tn.

1.8 Hierarchical routing results in shorter paths than flat routing

True.

because in hierarichal routing we check layer by layer. whereas in flat routing we go until the destination is found.

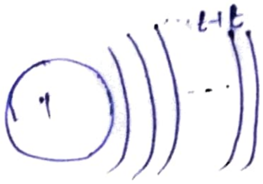
0

In hierarichal we first check the neighbours of source node then their neighbours so on. so this results in shorter paths.

1.9 A change in the service provided at layer "k", will impact services at both layers "k-1" and "k+1".

→ FALSE

in general, layer t can access the functionalities of layer below it (t-1, t-2, ----)



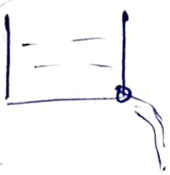
→ Thus no effect for layer k-1 (because layer k-1 can access ~~than~~ service below it but not above it.

→ Thus It will impact services at layer K+1.

1.10 “Leaky bucket algorithm” performs better than the “token bucket algorithm”.

true ✗

0



whereas token is based on ~~token~~ token i.e. when they get chance they dot

2. Answer the following briefly [5*3=15]
   2.1 Suppose that a multi-programmed system has a load of N processes with individual total execution times of $t_1, t_2, \ldots, t_N$. How would it be possible that total execution time could be as small as maximum $(t_1, t_2, \ldots, t_N)$?

→. let say t_N is

0

2.2 Explain the reason why transport protocols use larger size sliding windows as compared to the size of the size of sliding windows of data link layer?

→ The reason why transport protocols sliding windows use larger size as compare to size of sliding windows of data link layer because the. ~~po~~ PDU in transport layer is segment whereas in data link layer is is frame. so more number of segments to be sent as frames can be divided into ~~bata~~ frames ~~of~~ of smaller size

0

Delays

2.3 Discuss the following program threats: trap door, virus and denial of service.

0

2.4 What do you mean by admission control? Why not admission control is employed in a typical OS ?

admission control ~~in~~ ~~tp~~ means the program. gets control over cpu as soon as it enters. X

0.5

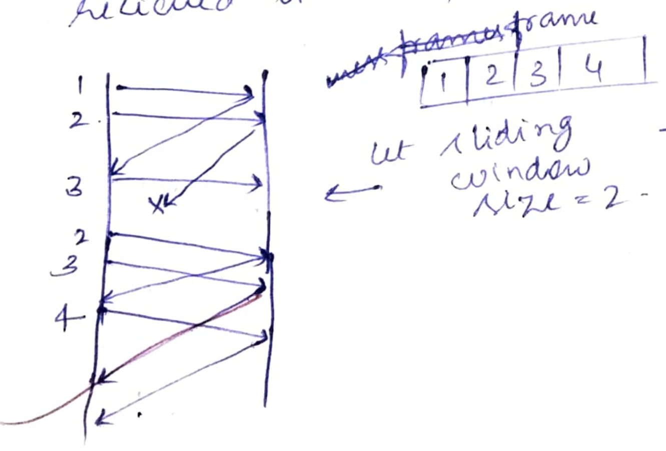
But this is not employed in O S because ~~the~~ if a new process with large burst time then cpu utilisation ↓. because lower burst time process are waiting which could be ~~soon~~ finished executing in cpu very fastly. so response time ↑. so this is not employed in typical O S

2.5 Explain the tradeoffs between "Go-Back-N" Sliding Window Protocol and "Selective Repeat" Sliding Window Protocol.

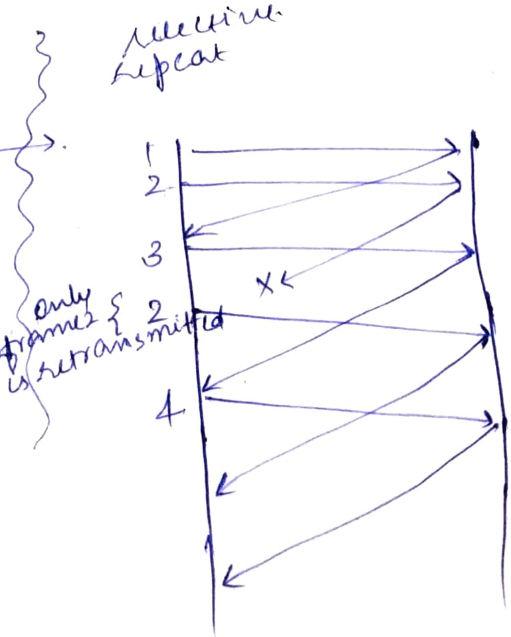
→. In go-back-N sliding window protocol if acknowledgement of ~~process~~ message is not received then all the messages in the window must be sent again (retransmitted)

2

But in selective repeat sliding window protocol only the message for which acknowledgment is not ~~send~~. received is only retransmitted.



If say acknowledgment packet of frame 2 isn't received within a specific interval of time then both ~~the~~ frames 2,3 sent again.



less number of retransmission possible here compared to go-back-N sliding window pro

3. The kernel of a multiprogramming system classifies a program as CPU-bound or I/O-bound and assigns appropriate priority to it. What would be the consequence of a wrong classification of programs for throughput and response times in a multi-programming system ? [5]

→ cpu bound: The process which uses cpu most of the time in its execution.

I/O bound: The process which uses I/O most of the time in its execution.

Now the processes are assigned priority ~~as~~ based on classification of cpu bound & I/O bound process.

let process A ← I/O bound process B ← cpu bound

In general. if at first I/O bound process A is taken into cpu. Now after execution of process A in cpu for a while it ~~goes~~ waits for I/O so it goes to waiting/blocked state. Now another process B is being run on cpu (say this cpu bound). Now both process gets executed simultaneously one in I/O one in cpu. Thus throughput and response time are very good in this case.

Thus while other processes running in I/O schedule a cpu bound process to run on cpu. Thus because of this processes are executed simultaneously which implies better throughput. Also Better response time because as soon as A goes for I/O B started executing on cpu. so response time less.

(5)

Now say. A (actually I/O bound) is classified as cpu bound process and B is (actually cpubound) classified as I/O bound process. Thus for better throughput the O.S schedules B first to run on cpu as it was classified as I/O bound. ~~This~~ Here O.s thinks that after I/O bound went to waiting state for execution ~~of I/O~~ in cpu bound process so that now they can run simultaneously we can schedule one in cpu and one in I/O. But ~~we~~ classification it was ~~that~~ is infact wrong. so B got to run first in cpu. As it cpu bound time it uses cpu most of the time. Now after completion of B, A is scheduled to run in cpu. As it is I/O bound, most of the time it uses I/O But ~~now~~ cpu is free (if no other processes are there). Thus cpu utilisation ↓. so throughput, response times also decreases because A runs on cpu after B's ~~[illegible]~~ got executed. so wrong classification of program may decrease throughput, responsetime.

4. Given that the monitor construct protects critical section by allowing one process at a time, why you require "c.wait" and "c.signal" operations in the Monitor. How they are different from "wait" and "signal" operations of semaphore?[5]

→ let s be a semaphore.

```
wait( s)
while (true).
{
    if (s <= 0) continue;
    else {s --; break}
}

signal(s).
    {
        s++;
    }
```

2.5

but these may cause deadlock. we know these are used to provide mutual exclusion. But improper code lead to mutual exclusion

so monitors are used.

here automatically the mutual exclusion problem is overcomed because it allows one process at time.

here there is no problem of deadlock because of modified wait, signal.

```
c.wait~~(s)~~
{   c--;
    if (c<0) { suspend (process)}
    ~~else s~~;
}
```

here instead of waiting the process is suspended and process pointer is enqueued

```
c.signal~~(s)~~
{   c++;
    ~~if~~ if (c<=0)
      resume (process);
    else ~~c++~~;
}.
```

the process in the dequeue gets resumed.

here instead of waiting the process is suspended & resumed. This avoid deadlock (indefinite waiting).

5. In addition to hardware support such as page table and secondary memory what kind of software support is needed to implement demand paging ? Explain clearly. [5]

6.

1

→

✓ there must be a memory management unit where the logical address generated by the process is converted to physical address via it. The support of this ~~is~~ very useful because it only identifies in which memory frame the corresponding page data is there.

✓ TLB (translation look aside buffer).

↳ in general in main memory, only page table is stored. so for accessing the data 2 times the main ~~pagetable~~ memory is ~~used~~ accessed.

so to decrease the time taken TLB introduced which store some pagetable entries.

— Handling page faults

$$\text{time} = \text{Hitratio} *.(t_{TLB}) + (1 - \text{Hitratio}) * (t_{TLB} + 2 t_{mm})$$

↗

— modebit .

the page.table entry of this is ~~stored~~ there in TLB

✓ Register PTBR (page table base registers) which stores the base address of page.table.

If page table size is small then it can also be stored in register then time to access data = tmm.

6. A computer has a cache, main memory, and a disk used for virtual memory. If a referenced word is in cache, 20 nsec (nano second) are required to access it. If it is in main memory but not in cache, 60 nsec are needed to load into the cache, and then the reference is started again. If the word is not in main memory, 12 msec (milli seconds) are required to fetch the word from the disk, followed by 60 nsec to copy it to the cache, and then the reference is started again. The cache hit ratio is 0.9 and the main memory hit ratio is 0.6. What is the average time, in nanoseconds, required to access a referenced word on this system ? [5]

5

→ time taken to access information in cache = 20 nsec. (t_c)

time taken to ~~access information~~ bring ~~informat~~ data from mainmemory to cache = 60 nsec (t_m)

time taken to bring data from disk to cache = 12 msec + 60 nsec (t_d).

avg (average) time required to access a referenced word on this system = cache hit * t_c + cachemiss * memory hit * $(t_m + t_c)$ + cachemiss * memorymiss * $(t_d + t_c)$.

* because after bringing into cache extra t_c time taken to access it

$\Rightarrow (0.9) * (20 \times 10^{-9}) + (1-0.9) * (0.6) * (80 \times 10^{-9})$
$+ (1-0.9) * (1-0.6) * (.12 \times 10^{-3} + 80 \times 10^{-9})$.

$\Rightarrow 18 \times 10^{-9} + 4.8 \times 10^{-9} + 3.2 \times 10^{-9} + 48 \times 10^{-5}$

$\Rightarrow$ ~~2~~ $(\overset{26}{\cancel{25.2}} \times 10^{-9} + 4.8 \times 10^{-5})$ seconds.

$\Rightarrow$ 480026 nsec

Ans

8·

7. Assume a system with four resource types C=<6,4,4,2>, and the maximum claim table shown below. The resource allocator is considering allocating resources according to the table shown below. Is this safe state ? Why and why not ? [5]

You can start the answer from here:

Maximum Claim table				
Process	R0	R1	R2	R3
P0	3	2	1	1
P1	1	2	0	2
P2	1	1	2	0
P3	3	2	1	0
P4	2	1	0	1

Current Allocation Table				
Process	R0	R1	R2	R3
P0	2	0	1	1
P1	1	1	0	0
P2	1	1	0	0
P3	1	0	1	0
P4	0	1	0	1
	5	3	2	2

→ resources ~~used~~ allocated in current allocation table = <5,3,2,2>

resources left = <6,4,4,2> - <5,3,2,2>
= <1,1,2,0>

resource still need to be allocated to process

process	R0	R1	R2	R3
P0	1	2	0	0
P1	0	1	0	2
P2	0	0	2	0
P3	2	2	0	0
P4	2	0	0	0

← ((3 2 1 1) - (2011))
max claim
current allocated.

*. We run. bankers safety algorithm to check if it possible to find safe sequence. If safe sequence exist then in safe state or not in safe state.

clearly P2 <0,0,2,0> $\boxed{<=}$ <1,1,2,0>
require resource
so we can allocate to P2. resources left = <1,1,0,0>

Now as all the resources for P2 are allocated it ~~can~~ release the resources now

the resources left = <2,2,2,0> because P2 releases <1,1,2,0>

Now required resources can be given to P0 or P3 or P4 let say given to P0

Now after P0 has completed it release the resources → resources left = <4,2,3,1>

Now P3 has been given and it release the resources upon completion resources left <5,2,4,1>

lly P4 → resources left <5,3,4,2>

lly P1 → resources left <6,4,4,2> ↳ finally all the resources are free.

Now we found a safe sequence = < P2 P0 P3 P4 P1 > ~~proved~~!

Thus it is in safe state

1

8. Consider three sets of users, professor, TA and student. There exists a folder marks to which only professor can write into and student and TA can only read from. There exists another folder called course material which can be written to by professor and TA and students can only read from it. Use Access Control Matrix, determine a security policy for this criteria. [5].

→ Access control Matrix

mark → F1, course material → F2

	F1	F2
professor	W/R	W/R
TA	R	W/R
student	R.	R

} owner } group.

2+2=4

The folder f1 (marks) can be accessed only by professor so security ~~was~~ must be provided such that TA, student cant access it.

||ly The folder F2 (course material) can be accessed. ~~only~~ by. students, professor and student cant access it security must be provided to it such that student cant access it.

- for this we first group all professor under a category, all TA's into another category, all. student into other category.

- for this for folder (f1) we allow only ~~user~~ owner to access it. so enable permission only for the owner.

for (F2) we create a group which include. TA, ~~prof~~ thus now for this folder write permission are given to group owner

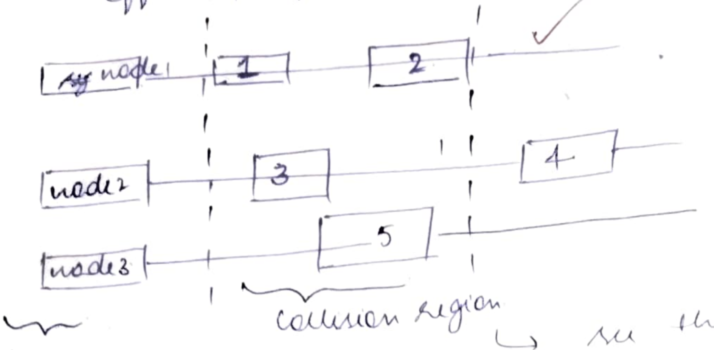
In this way security is ensured. ~~for~~ via. access control matrix.

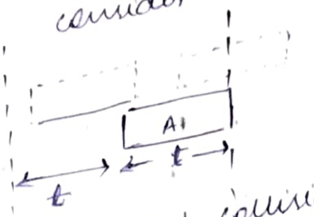
9. Explain the difference between "Pure ALOHA" and "Slotted ALOHA" in terms of implementation? What is the reason for the improved performance of "Slotted ALOHA" over "Pure ALOHA" [5]

→ pure aloha: the nodes can send information whenever they want to send

But this may lead to ~~intra~~ collisions Because of this efficiency of pure aloha ≃ 18%

consider equal size of frames





to avoid collision with A there must be no message sent in duration of 2t probability of this = P₁ (say) —(1)

Here three nodes communicate via same link.

→ so there are so many collisions occurring here Because of this data is lose / corrupted. so nodes have to retransmit the ~~are~~ frames, ~~to~~ after ~~waiting~~ a random ~~energy~~ amount of time so as to ensure that ~~no collision~~ same collisions dont occur

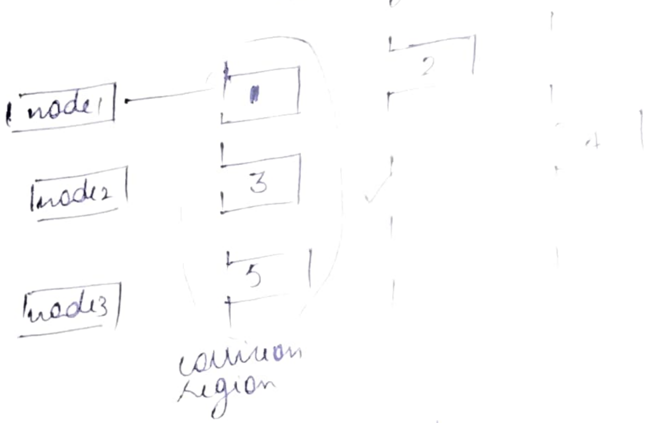
5

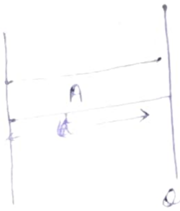
As the number of ~~colli~~ collisions are on aloha (pure) due to efficiency of message successfully gets transmitted to receiver is only ~~18%~~ nearly 18%.

so that why slotted aloha was found here the number of collisions decreased due to which efficiency ~~ss~~ of message successfully gets transmitted to receiver is nearly 37%. performance get doubled

slotted aloha: here the channel is divided into time slots and the message must be transmitted only at the beginning of the slot

good!





as the message frame was sent only at beginning of the time slot so thus there must be no message in duration of t probability of this = P₂ (say) —(2)

from (1) (2) we conclude $P_1 < P_2$. probability that no message sent in duration of 2t < probability that no message sent in duration of t

⇒ collision in aloha > collisions in slotted aloha
Thus improved performance of slotted aloha over pure aloha.

10. Explain the issues of connection establishment in the Transmission Control Protocol of transport layer? How three-way handshake solve the problem. [5]

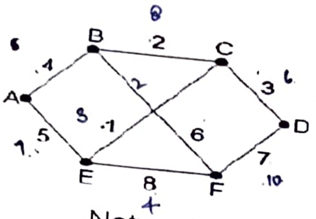
→ There are two transport layer protocols : TCP
UDP

TCP - ~~Transmission control protocol~~
UDP - user datagram protocol.

The data rate in UDP is very fast but its not reliable. The data gets lost here. So we think TCP protocol is good. But so it also has issues. It ~~[illegible]~~advantage : its reliable (no data loss). It disadvantage is low data rate. And there are many more problems in TCP which could be solved by three way hand shake.

Generally Transmission control protocol has less issues as compared to UDP because in UDP data loss which is the main problem is not a problem in TCP. Here the data rate is a bit slow as compared to Q10. Also there are many more problems in TCP which were. solved by three way handshake. Thus. three way handshake is a good solution to transport layer. for TCP.

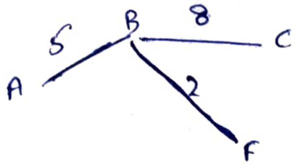
11. Consider the following network. Distance vector routing is used and the following vectors have just come in to router C; from B: (5,0,8,1,6,2); from D: (16,12,6,0,9,10); from E (7,6,3,9,0,4). The cost of the links from C to B, D, and E, are 6, 3 and 5 respectively. What is C's new routing table? [5]



Network

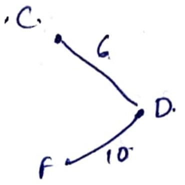
→ A B C D E F

B (5 0 8 1 6 2)



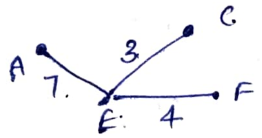
A B C D E F

11y D (16 12 6 0 9 10)

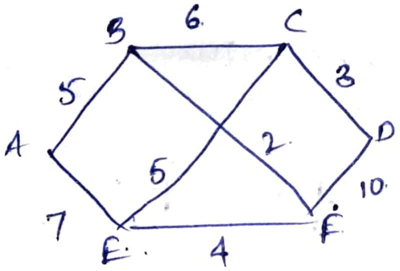


A B C D E F

E (7 6 3 9 0 4)



Now $C \overset{6}{-} B$ $C \overset{3}{-} D$ $C \overset{5}{-} E$



C(11, 6, 0, 3, 5, 8)

5

B

12. Jurassic Park consists of a dinosaur museum and a park for safari riding. There are "m" passengers and "n" single-passenger cars. Passengers wander around the museum for a while, then line up to take ride in a safari car. When a car is available, it loads the one passenger it can hold and rides around the park for a random amount of time. If the "n" cars are all out riding passengers around, then the passenger who wants to ride waits; if a car is ready to load but there are no waiting passengers, then the car waits. Use semaphores to synchronize the "m" passenger processes and the "n" car processes. Is your solution deadlock and/or starvation free ? Discuss. [5]

→ let mutex cars = n; ← counting semaphores.

```
passenger process ÷ for all m processes.
while (true)
{
    // passenger wander around museum for random amount of time
    wait (cars);
    // (ride around the park for random amount of time).
    signal (cars);
}
```

Here the passenger roams for a while and then he waits for the car if ~~none~~ none of them arent free. else he goes into car.
↓
possible iff n<m.

– Here there is no deadlock because here the condition hold and wait for the deadlock isn't satisfied. because passengers are waiting only for 1 car. so no hold and wait ⇒ No deadlock.

– Here there is ~~no~~ starvation ~~because~~ possible iff n<m. because all the passengers are only continuously using the car. so there may be starvation. Below is the final code to avoid starvation.

Thus the final code ÷ let condition[m]; all values are false.
number[m]; all values are initialised to zero.
mutex cars = n;

5

```
passanger i
while (true)
{
    condition[i] = true;
    number[i] = max(number[0], number[1], ... number[m-1]) + 1;
    condition[i] = false;
    ~~wait (cars);~~
    for (int j = 0; j < m; j += 1)
    {
        while (condition[j]); ~~{ wait (cars)~~;
        while (number[j] != 0 && number[j, j] < number[i, i]
        ~~wait (cars); signal (cars); }~~
    }
    wait(cars);
    // drive the car for random amount of time
    number[i] = 0;
    signal (cars);
}.
```

here (a, c) < (b, d)
⇒ a < b or (a == b &&
c

Thus the passengers are given cars in order
thus now no starvation!
No deadlock because no hold and wait condition satisfied.
Thus it is deadlock & starvation free!

13. Developments in operating systems have generally occurred in an evolutionary rather than revolutionary fashion. For each of the following transitions, describe the primary motivations of operating systems designers that led them to produce the new type of system from the old. [5]

13.1 Single user dedicated systems to multiprogramming.

13.2 Fixed partition multiprogramming systems with absolute translation and loading to fixed partition multiprogramming systems with re-locatable translation and loading.

13.3 Fixed partition multiprogramming to variable partition multiprogramming.

13.4 Contiguous storage allocation systems to non-contiguous storage allocation systems.

13.5 Single user dedicated systems with manual job-to-job transition to single user dedicated systems with single stream batch processing systems.

13.4.

→ because in contiguous storage allocation, internal fragmentation is a prollem in fixed partition and external fragmentation in variable partition. so. memory wastage there. Thus. non-contiguous storage allocation was introduced.

13.1

→ here if the ~~pr cpu~~ process is I/O bound then cpu is free. Thus cpu utilisation is less here. so multiprogramming came.

1

13.3

→ If the program size less than partition then there will be memory wastage. so variable partitioning came.

1

13.2

→ instead of loading complete program loading some part of program and then other. enhance interactiveness

~~so~~

13.5

→ instead of manual job to job transition ~~it less~~ evolution to. ~~inefficient then~~ single stream batch processing systems because. batch-background processing systems. they ~~execut~~ have the lowest priority and they get executed. whenever no foreground / no. system program are there. here. time slicing can be used where batch processing systems executed with low priority.