Q1)

Consider the following snapshot of a system:

| | Allocation | Max | Available |
|---|---|---|---|
| | A B C D | A B C D | A B C D |
| $T_0$ | 0 0 1 2 | 0 0 1 2 | 1 5 2 0 |
| $T_1$ | 1 0 0 0 | 1 7 5 0 | |
| $T_2$ | 1 3 5 4 | 2 3 5 6 | |
| $T_3$ | 0 6 3 2 | 0 6 5 2 | |
| $T_4$ | 0 0 1 4 | 0 6 5 6 | |

Answer the following questions using the banker's algorithm:

a. What is the content of the matrix ***Need***?

b. Is the system in a safe state?

c. If a request from thread $T_1$ arrives for (0,4,2,0), can the request be granted immediately?

**Answer:**

a. The values of ***Need*** for processes $P_0$ through $P_4$, respectively, are (0, 0, 0, 0), (0, 7, 5, 0), (1, 0, 0, 2), (0, 0, 2, 0), and (0, 6, 4, 2).

b. The system is in a safe state. With ***Available*** equal to (1, 5, 2, 0), either process $P_0$ or $P_3$ could run. Once process $P_3$ runs, it releases its resources, which allows all other existing processes to run.

c. The request can be granted immediately. The value of ***Available*** is then (1, 1, 0, 0). One ordering of processes that can finish is $P_0$, $P_2$, $P_3$, $P_1$, and $P_4$.

For part b) the possible **termination sequence** would be ***p3,p0,p1,p4,p2***

- if we assume (for example, question had different Allocation and Max matrices) that once process p0 or p3 run and other existing processes cannot run, then the **termination sequence** would be ***p3,p0, Deadlock***

Q2)

Consider the following snapshot of a system:

| | Allocation | Max |
|---|---|---|
| | A B C D | A B C D |
| $T_0$ | 3 0 1 4 | 5 1 1 7 |
| $T_1$ | 2 2 1 0 | 3 2 1 1 |
| $T_2$ | 3 1 2 1 | 3 3 2 1 |
| $T_3$ | 0 5 1 0 | 4 6 1 2 |
| $T_4$ | 4 2 1 2 | 6 3 2 5 |

Using the banker's algorithm, determine whether or not each of the following states is unsafe. If the state is safe, illustrate the order in which the threads may complete. Otherwise, illustrate why the state is unsafe.

a. ***Available*** $= (0, 3, 0, 1)$

b. ***Available*** $= (1, 0, 0, 2)$

Answer)

a. Not safe. Processes $P_2$, $P_1$, and $P_3$ are able to finish, but no remaining processes can finish.
Safe. Processes $P_1$, $P_2$, and $P_3$ are able to finish. Following this, processes $P_0$ and $P_4$ are also able to finish.

For part a) **Termination sequence** is : *P2,p1,p3,Deadlock*

Q3) Fill in the blanks

A --- address does not refer to an actual physical address; rather, it refers to an abstract address in an abstract address space. A --- address refers to an actual physical address in memory. A --- address is generated by the CPU and is translated into a physical address by ---.

Answers, respectively) logical, physical, logical, the memory management unit(MMU)

Q4)

Consider a logical address space of 64 pages of 1,024 words each, mapped onto a physical memory of 32 frames.

a. How many bits are there in the logical address?

b. How many bits are there in the physical address?

Answer)

a. Logical address: 16 bits

b. Physical address: 15 bits

Q5)

Given six memory partitions of 300 KB, 600 KB, 350 KB, 200 KB, 750 KB, and 125 KB (in order), how would the First-fit, best-Fit, and worst-Fit algorithms place processes of size 115 KB.

Answer)

**First fit)** 115 KB is put in 300-KB partition, leaving 185 KB, 600 KB, 350 KB, 200 KB, 750 KB, 125 KB

**Best fit)** 115 KB is put in 125-KB partition, leaving 300 KB, 600 KB, 350 KB, 200 KB, 750 KB, 10 KB

**Worst fit**) 115 KB is put in 750-KB partition, leaving 300 KB, 600 KB, 350 KB, 200 KB, 635 KB, 125 KB

Q6)

Consider a computer system with a 32-bit logical address and 4-KB page size. The system supports up to 512 MB of physical memory. How many entries are there in each of the following?

a. A conventional, single-level page table

b. An inverted page table

Answer)

a) Page offset(size) = ($2^2$ x $2^{10}$)= frame offset(size)

32-12 =20

a. $2^{20}$ entries.

b. 512 K K/4K = 128K entries.

Q7)

Under what circumstances do page faults occur? Describe the actions taken by the operating system when a page fault occurs.

**Answer:**

A page fault occurs when an access to a page that has not been brought into main memory takes place. The operating system verifies the memory access, aborting the program if it is invalid. If it is valid, a free frame is located and I/O is requested to read the needed page into the free frame. Upon completion of I/O, the process table and page table are updated, and the instruction is restarted.

Q8)

Consider the following page reference string:

**7, 2, 3, 1, 2, 5, 3, 4, 6, 7, 7, 1, 0, 5, 4, 6, 2, 3, 0 , 1.**

Assuming demand paging with three frames, how many page faults
would occur for the following replacement algorithms?

- LRU replacement
- FIFO replacement
- Optimal replacement

**Answer:**

- 18
- 17
- 13

Q9)

Matching question)

Consider a demand-paged computer system where the degree of multi- programming is currently fixed at four. The system was recently measured to determine utilization of the CPU and the paging disk. Three alternative results are shown below. For each case, what is happening? Can the degree of multiprogramming be increased to increase the CPU utilization? Is the paging helping?

a. CPU utilization 13 percent; disk utilization 97 percent

b. CPU utilization 87 percent; disk utilization 3 percent

c. CPU utilization 13 percent; disk utilization 3 percent

Answer to match:

1. CPU utilization is sufficiently high to leave things alone and increase the degree of multiprogramming.

2. Increase the degree of multiprogramming.

3. Thrashing is occurring.

**a→3**

**b→1**

**c→2**

Q10)

Which of the following is NOT part of the standard solution to the critical section problem:

a. Mutual exclusion.
**b**. Assumption regarding relative execution speeds of processes.
c. Progress.
d. None of the above.

Q11)

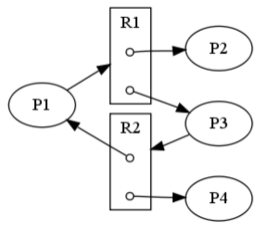
What is wrong with the following dining philosopher logic:

```
do {
    wait (chopstick[i] );
     wait (chopStick[ (i + 1) % 5] );

              //  eat

     signal (chopstick[i] );

               //  think

} while (TRUE);
```

a. When the two chopsticks are in use, the process does not wait properly.
**b.** The process fails to release the chopsticks properly.
c. The process doesn’t try to access the proper chopsticks.
d. Nothing is wrong with the logic.

Q12)

The following resource graph has:




a. Cycle and deadlock.
b. Deadlock but no cycle.
c. Cycle but no deadlock.
d. No deadlock and no cycle.

Answer) C

Q and a 13)

Consider a memory containing 16 blocks, with the allocation described by the following Inverted Page Table (An entry contains Process number and block number within the process:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1,4 | P2,2 | P2,4 | P1,1 | P1,0 | P2,3 | P3,1 | P1,3 | P2,1 | | P1,5 | P3,0 | | P2,0 | P1,2 | |

From the above Inverted Page Table, complete the Page tables for Processes P1 and P2 below.

| **P1** | |
|---|---|
| **Frame #** | **Valid (1/0)** |
| **4** | 1 |
| **3** | 1 |
| **14** | 1 |
| **7** | 1 |
| **0** | 1 |
| **10** | 1 |

| **P2** | |
|---|---|
| **Frame #** | **Valid (1/0)** |
| **13** | **1** |
| **8** | **1** |
| **1** | **1** |
| **5** | **1** |
| **2** | **1** |
| **-** | **0** |

**Note that P2 does not use as many pages as P1. In fact, if you examine process P3, you will see that only 2 frames have been allocated to this process.**

Q and Answer 14)

Provide a short definition for each of the following terms:

Page fault:

**A page fault occurs when the CPU tries to use a logical address for which a corresponding physical address does not exist, that is, the process page that contains the logical address had not been loaded into physical memory. The OS must load the logical page into physical memory to allow the process to continue its execution.**

Resident Set

**The set of physical frames occupied by a process (note that the resident set cannot exceed the physical frames allocated to the process).Another definition: the set of logical pages of a process loaded into physical memory. Both definitions are essentially equivalent – just two different perspectives.**

Working Set:

**The set of pages required by a process to execute, that is, according to locality of reference, the set of pages required by the process in memory to keep the number of page faults to a minimum.**

Q15) CPU scheduling algorithm

## Part 3: Problem Solving (24 points)

**Question 1. (9 points – 3 each)** Suppose that the following processes arrive the times indicated to run on the CPU for the indicated burst times.

| Process | Arrival Time | Burst Time |
|---|---|---|
| P1 | 0.0 | 16 |
| P2 | 1.5 | 2 |
| P3 | 1.0 | 1 |

a) What is the average turnaround time for these processes when using the FCFS (first-come first-serve) CPU scheduling algorithm?

FCFS

0 P1 16 ←P1 P3 17 ←P3 P2 19 ← P2

$w_{p_1} = 16 - 0 - 16 = 0$

$w_{p_2} = 19 - 1.5 - 2 = 15.5$

$w_{p_3} = 17 - 1 - 1 = 15$

$\bar{w} = \frac{0 + 15.5 + 15}{3} = 10.16$ average waiting time

$\bar{T} = \bar{w} + \left(\frac{16+2+1}{3}\right) = 16.49$

average turnaround time

b) What is the average turnaround time for these processes when using the SJF (shortest job first) with **preemption** CPU scheduling algorithm?

SRJF

SRTF

0 P1 1 P3 2 P2 4 P1 19 ← exit time P1

exit time P3

exit time P2

$w_{p_1} = 19 - 0 - 16 = 3$

$w_{p_2} = 4 - 1.5 - 2 = 0.5$

$w_{p_3} = 2 - 1 - 1 = 0$

$\bar{w} = \frac{3 + 0.5 + 0}{3} = 1.16$

$\bar{T} = \bar{w} + \left(\frac{16+2+1}{3}\right) = 7.49$

c) What is the average turnaround time for these processes when using the round robin CPU scheduling algorithm with a quantum of 3?

R.R.

0 P1 3 P3 4 P2 6 P1 19 ← P1

P3

P2

$w_{p_1} = 19 - 0 - 16 = 3$

$w_{p_2} = 6 - 1.5 - 2 = 2.5$

$w_{p_3} = 4 - 1 - 1 = 0$

$\bar{w} = \frac{3 + 2.5 + 0}{3} = 1.83$

$\bar{T} = \bar{w} + \left(\frac{16+2+1}{3}\right) = 8.16$

Extra example

p1 exit　　p3 exit　　p2 exit

sjf (n-pre)

0 p1 16 p3 17 p2 19

$w_{p1} = 16 - 0 - 16 = 0$

$w_{p2} = 19 - 1.5 - 2 = 15.5$

$w_{p3} = 17 - 1 - 1 = 15$

$\overline{w} = \frac{0 + 15.5 + 15}{3} = 10.16$

$\overline{w} + \left( \frac{16+2+1}{3} \right) = 16.49$

average waiting time $\overline{w} = \frac{\sum_{i=1}^{n} w_{p_i}}{n}$

average turnaround time $= \overline{w} + \left( \frac{\sum_{i=1}^{n} \text{cpu burst } p_i}{n} \right)$