National University of Computer and Emerging Sciences, Lahore Campus

Course Name:	Operating Systems	Course Code:	CS2006
Degree Program:	BS (CS / SE / DS)	Semester:	Spring 2023
Exam Duration:	180 Minutes	Total Marks:	90
Paper Date:	08-June-2023	Weight	45%
Section:	ALL Sections	Page(s):	11
Exam Type:	Final Exam		

Student : Name:_______________________________ **Roll No.**_________________ **Section:**________

Instruction/Notes: Avoid unnecessarily explanation. Kindly write your information in the above mentioned space. Rough sheet is also attached on the last page. No extra sheet is allowed.

CLOs	CLO-3	CLO-3	CLO-5	CLO-2	CLO-5	CLO-5	
Questions	Q-1	Q-2	Q-3	Q-4	Q-5	Q-6	Total
Total Marks	10	20	15	10	15	20	**90**
Marks Obtained							

Question 01: (10 points) (CLO-3)

Javed, Yusuf, and Babar go to a restaurant at a busy time of the day. The waiter apologetically explains that the restaurant can provide only two pairs of spoons (for a total of four spoons) to be shared among the three people. Yusuf proposes that all four spoons be placed in an empty glass at the center of the table and that each diner should obey the following protocol:

```
semaphore spoon = 4;

while (!had_enough_to_eat())
{
        wait(spoon);
        wait(spoon);
        eat();
        signal(spoon);
        signal(spoon);
}
```

(a) Can this dining plan lead to a deadlock? YES or NO. Explain your answer.

Answer:

(b) Suppose now that instead of three there will be an arbitrary number of *D* diners. Furthermore, each diner *d* = 1...*D* may require a different number of S_d spoons to eat. For example, it is possible that one of the diners is an octopus, who for some reason refuses to begin eating before acquiring *octopus* = 8 spoons. For example, Javed, Yusuf, Babar, and one octopus would result in C = 14. Each diner's eating protocol will be as displayed below:

```
int s;
int num_spoons = my_spoon_requirement();
while (!had_enough_to_eat())
{
        for (s = 0; s < num_spoons; s++)
        {
                wait (spoon); /* May block. */
        }
        eat ( );
        for (s = 0; s < num_spoons; s++)
        {
                signal (spoon); /* Does not block. */
        }
}
```

What is the smallest number of spoons (in terms of D and S_d) needed to ensure that deadlock cannot occur? Explain your answer.

Answer:

Question 02: (20 points) (CLO-3)

Consider a Multilevel Feedback Queue Scheduler having three queues numbered from 1 to 3 (see Fig. A). The processes are scheduled as follows:

- A new process enters queue 1 which is served using Round Robin (RR). When it gains CPU, process receives 8 milliseconds. If it does not finish in 8 milliseconds, process is moved to the end of queue 2.
- If queue 1 is empty, the processes at queue 2 are served using RR and receives 16 milliseconds. If it does not complete, it is preempted and moved to queue 3.
- Processes in queue 3 are run on a First Come First Serve (FCFS) basis, but are run only when queues 1 and 2 are empty.
- A process that arrives for queue 2 will preempt a process in queue 3. A process in queue 2 will in turn be preempted by a process arriving for queue 1.
- If a process does not use up its quantum in queue 2 due to preemption by queue 1, it will keep its current queuing level and be put into the end of the queue. Then, it can still get the same amount of quantum (not remaining quantum) next time when it is picked.

The following set of processes, with the arrival times and the length of the CPU-burst times given in milliseconds, have to be scheduled using this Multilevel Feedback Queue Scheduler:

Process	Arrival Time	Burst Time
P1	0	17
P2	12	25
P3	28	8
P4	36	32
P5	46	18

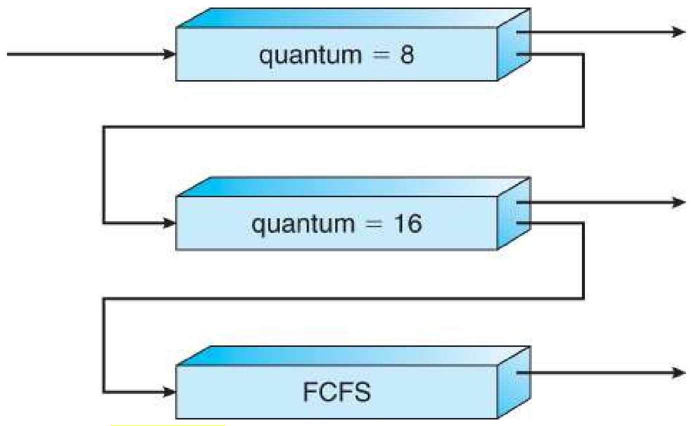


Figure A: Multilevel feedback queue

(a) Draw a Gantt chart illustrating the execution of these processes.

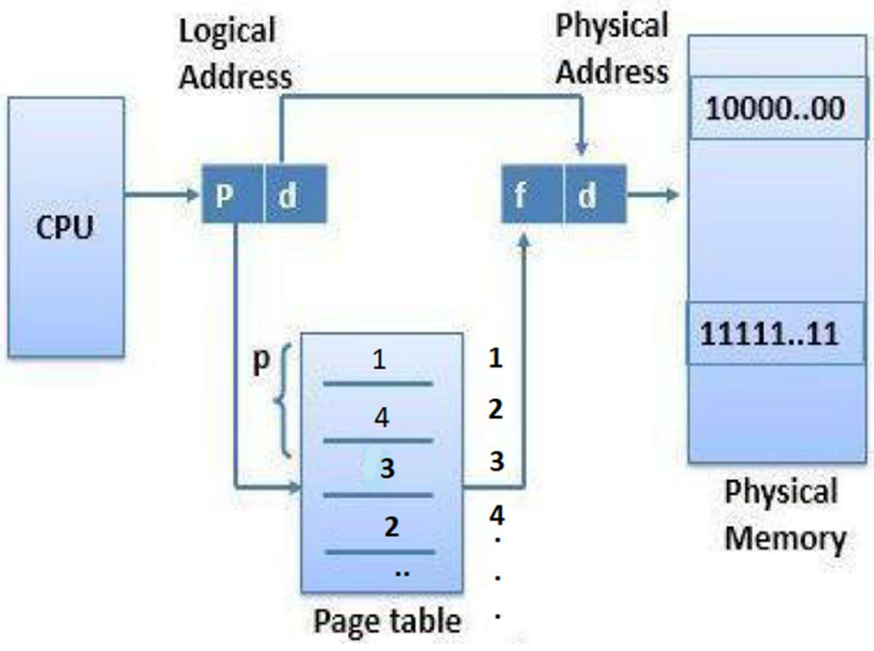
(b) Calculate the number of context switches for the processes.

(c) Calculate the average waiting time and the average turnaround time for the scheduling.

Question 03: (15 points) (CLO-5)

Consider a system with **20** bits logical address and **4 MBs** main memory size and page size is **4 KBs**.

1200 is logical address



A. Convert 1200 into its binary representation

B. No. of bits needed for p (page number)

C. No. of bits needed for f (frame number)

D. No. of bits needed for d (offset)

E. Convert logical address into physical address (you have to provide its binary as well as decimal representation).

Question 04: (10 points) (CLO-2)
Write the output of the following code.

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

int main( ) {
   int n = 5;
   pid_t pid;

   for (int i = 0; i < n; i++) {
      pid = fork( );

      if (pid == 0) {
         if (i % 2 == 0) {
            fork( );
            printf ("Child process %d\n", i);
         } else {
            printf ("Child process %d\n", i);
            fork( );
         }
         break;
      } else if (pid > 0) {
         wait (NULL);
         printf ("Parent process\n");
      } else {
         printf ("Fork failed\n");
         return 1;
      }
   }

   return 0;
}
```

Answer:

Question 05: (15 points) (CLO-5)
A process has four page frames allocated to it. (All the following numbers are decimal, and everything is numbered starting from zero). The time of the last loading of a page into each page frame, the time of last access to the page in each page frame, the virtual page number in each page frame, and the referenced (R) and modified (M) bits for each page frame are as shown (the times are in clock ticks from the process start at time zero to the event -- not the number of ticks since the event to the present).

Virtual page number	Page frame	Time loaded	Time referenced	R bit	M bit
2	0	60	164	1	1
1	1	30	166	1	0
0	2	150	162	0	1
3	3	20	163	1	1

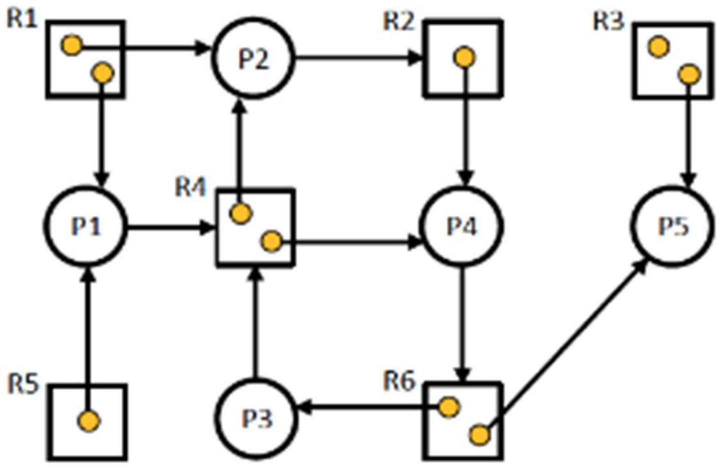
A page fault to virtual page 4 has occurred. Which page frame will have its contents replaced for each of the following memory management policies? Explain why in each case.

a) FIFO (first-in-first-out)

b) LRU (least recently used)

c) Optimal page replacement algorithm

Question 06: (20 points) (CLO-5)
Consider the following Resource Allocation Graph (RAG):



Do the following problems:

(a) Convert it to the matrix representation (i.e., Allocation, Request and Available)

(b) Is there a deadlock? If there is a deadlock, which processes are involved? Otherwise, provide safe sequence.

(Rough Sheet)