National University of Computer and Emerging Sciences 0046

Islamabad Campus

Operating Systems (CS2006)

Final Exam

Date: May 22, 2024

Total Time (Hrs): 3
Total Marks: 100
Total Questions: 7

Course Instructor(s):
Dr. M. Faisal Cheema, Dr. Adnan Tariq, Ms. Maryam Shahbaz, Ms. Rabail Zahid, Mr. M. Aadil-ur-Rehman

Roll No **Course Section** **Student Signature**

IMPORTANT: Attempt Question 1, 2 on Answer Sheet, and remaining Question on Question Paper. If you do not follow the instructions, your questions will not be marked.

Attempt all the questions.

Question 1: (12 Marks)
[Attempt this Question on the Answer Sheet]

Danyal, Sohaib, and Fahad plant seeds continuously. Danyal digs the holes. Sohaib then places a seed in each hole. Fahad then fills the hole up. There are several synchronization constraints:

1. Sohaib cannot plant a seed unless at least one empty hole exists, but Sohaib does not care how far Danyal gets ahead of Sohaib.
2. Fahad cannot fill a hole unless at least one hole exists in which Sohaib has planted a seed, but the hole has not yet been filled. Fahad does not care how far Sohaib gets ahead of Fahad.
3. Fahad does care that Danyal does not get more than MAX holes ahead of Fahad. Thus, if there are MAX unfilled holes, Danyal has to wait.
4. There is only one shovel with which both Danyal and Fahad need to dig and fill the holes, respectively.

Write the pseudocode for the 3 processes which represent Danyal, Sohaib and Fahad using semaphores as the synchronization mechanism. **[3 + 3+ 3 Marks]**
Make sure to initialize the semaphores correctly **[3 Marks]**

Question 2: (10 Marks)
[Attempt this Question on the Answer Sheet]

Canada and US are separate threads executing their respective procedures shown below. The code below is intended to cause them to forever take turns exchanging insults through the shared variable X in strict alternation. The Sleep() routine blocks the calling thread, and the Wakeup() routine unblocks a specific thread if that thread is blocked.

// Initializing Semaphores
creatableHoles = MAX // Danyal's sem
emptyHoles = 0 // Sohaib's sem
fillableHoles = 0 // Fahad's sem
shovel = 1 // binary sem that Danyal & Fahad

Danyal()
    wait(creatableHoles)
    while(!shovel);
    shovel = 0 // Crit. sec
    signal(emptyHoles) // Digging Hole
    shovel = 1 // end of C.S

Sohaib()
    wait(emptyHoles)
    signal(fillableHoles)

Fahad()
    wait(fillableHoles)
    while(!shovel)
    shovel = 0
    signal(creatableHoles)
    shovel = 1

ok at these 3
sleep();
x = showInsult
wakeup(US Th

threads can
to miniscule
cannot be
in both US a
sequentially.
It is very
runs wake
run both
the US The
resulting
time

b) S

Look at these 3 lines:

sleep();	x = shoutinsult(x);
x = shoutinsult(x);	wakeup(ThreadCanada);
wakeup(USThread);	sleep();

As threads can execute at different speeds due to miniscule changes in the CPU's speed, we cannot be 100% sure that the lines in both US and Canada codes will execute sequentially.
It is very likely that after the USC) thread runs wakeup(ThreadCanada), the Canada() thread may run both ShoutInsult() and wakeup(USThread) before the USThread can go to sleep (or vice versa), resulting in both threads sleeping at the same time and causing a deadlock.

b)
```
Semaphore s=1;
void Canada(){
        while(1){
            wait(s);
            x=ShoutInsult(x);
            signal(s);
        }
}
void USC){
        while (1){
            wait(s);
```

7.5

Q/Part No.

```
X = ShoutInsult(X);
signal(s);
    }
}
```

```
void
Canada ()
{
  while (1) {
    Sleep();
    X = ShoutInsult(X);
    Wakeup(USThread);
  }
}
```

```
void
US()
{
  while (1) {
    X = ShoutInsult(X);
    Wakeup(ThreadCanada);
    Sleep();
  }
}
```

a. The code shown above exhibits a well-known synchronization flaw. Outline a scenario in which this code would fail, and the outcome of that scenario. What is this flaw called? **[4 Marks]**

b. Show how to fix the problem using semaphores, replacing the Sleep() and Wakeup() calls with semaphore Wait (P/down) and Signal (V/up) operations. Note: Disabling interrupts is not an option. **[6 Marks]**

Question 3: (15 marks)

[Write your answer on the Question Paper only in the provided space below]

Consider the following page reference string:

1, 2, 3, 4, 2, 1, 5, 6, 2, 1, 3, 7, 6, 3, 2, 1, 2, 3, 6

Now assume that the total number of frames in main memory are 5. Fill the tables below using the FIFO,OPTIMAL and LRU page replacement algorithms for the given memory by filling in the table with page numbers. The top row is the string of memory references and each row contains the page numbers that resides in each physical frame F1-F5 after each memory reference.

Note: 1) If two or more frames are empty, the lower frame number is filled first. 2) If more than one page can be replaced according to the given policy, the page in the lower frame number will be replaced.

FIFO: //Assuming that lower frame number means FSCF1 (it doesn't make a difference running time).

	1	2	3	4	2	1	5	6	2	1	3	7	6	3	2	1	2	3	6
F1							5	5	5	5	5	5	5	5	2	2	2	2	2
F2				4	4	4	4	4	4	4	4	4	4	3	3	3	3	3	3
F3			3	3	3	3	3	3	3	3	3	7	7	7	7	7	7	7	7
F4		2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1
F5	1	1	1	1	1	1	1	6	6	6	6	6	6	6	6	6	6	6	6
	F	F	F	F	H	H	F	F	H	F	H	F	H	F	F	H	H	H	H

LRU:

	1	2	3	4	2	1	5	6	2	1	3	7	6	3	2	1	2	3	6
F1							5	5	5	5	5	7	7	7	7	7	7	7	7
F2				4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	3
F3			3	3	3	3	3	6	6	6	6	6	6	6	6	6	6	6	6
F4		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
F5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	F	F	F	F	H	H	F	F	H	H	F	F	H	H	H	H	H	H	H

OPTIMAL:

	1	2	3	4	2	1	5	6	2	1	3	7	6	3	2	1	2	3	6
F1							5	~~5~~	~~5~~	~~5~~	~~5~~	7	7	7	7	7	7	7	7
F2				4	4	4	4	~~6~~	~~6~~	~~6~~	~~6~~	6	6	6	6	6	6	6	6
F3			3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
F4		2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
F5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	F	F	F	F	H	H	F	F	H	H	H	F	H	H	H	H	H	H	H

Here, Replaced 6 in F2 with 6 because its the lower ~~order~~ page frame according to my assumption.

	Page Fault Ratio	Page Hit Ratio
FIFO	10/19=52.6%.	9/19=47.3%.
LRU	8/19=42.1%	11/19=57.9%.
OPTIMAL	7/19= 36.8%.	12/19=63.1%.

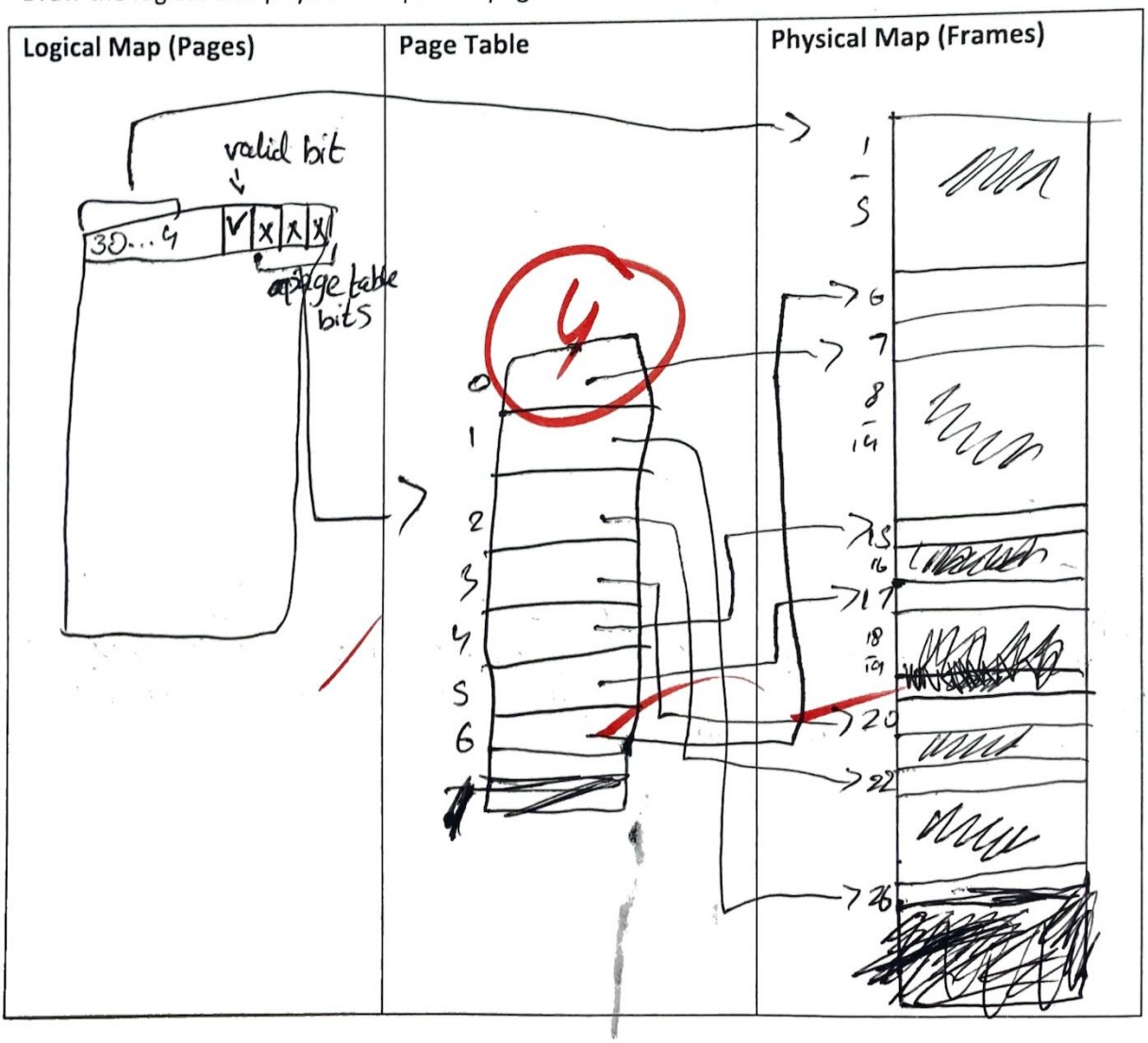
15

Question 4: (10 Marks)

[Write your answer on the Question Paper only in the provided space below]

Consider a user program of logical address of size 7 pages and page size is 4 bytes. The physical address contains 30 frames. The user program consists of 26 instructions a, b, c, . . . y, z . Each instruction takes 1 byte. Assume at that time the free frames are 7, 26, 22, 20, 15, 17 and 6. The first page will be kept at frame 7, the second page will be kept at 26, and so on.

Draw the logical and physical maps and page tables in the space. **[4 marks]**



The physical address of **j** [1 mark]	0000000 0000000 0001 0000000 000
The physical address of **o** [1 mark]	0000000000001 0000000 0000000
The physical address of **s** [1 mark]	0000000 0000000000000 000000
The physical address of **x** [1 mark]	0010 0000000 0000000000000
Amount of internal fragmentation *(in bytes)* **[2 marks]** Solution: 2 Bytes	6 bytes

Process #
P0
P3
P4
P1
P2
P3
P4
P8
P9
P8
P11

Question 5: (15 Marks)
[Write your answer on the Question Paper only in the provided space below]

The *Bakery Algorithm* for "n process mutual exclusion" is shown below.

```
boolean choosing[ n ];
int  number[ n ];
1  do {
2       choosing[ i ] = true;
3       number[ i ] = max(number[0], number[1], ..., number [n – 1])+1;
4       choosing[ i ] = false;
5
6       for (j = 0; j < n; j++) {
7               while (choosing[ j ]) ;
8               while ( (number[ j ] != 0) && (number[ j ], j ) <  (number[ i ], i ) )) ;
9       }
11              critical section
13       number[ i ] = 0;
14
15              remainder section } while (1);
```

A. We know that Bakery Algorithm does not guarantee that two processes do not receive the same token. In case of a tie, the process with the lowest ID is served first. Keeping this in mind, what is the purpose of **choosing** in lines 2 , 4 and 7? If we remove lines 2,4 and 7, will the code work fine. If so how? **[3 marks]**

Choosing ensures that we don't attempt to access the number of a specific process while it is ~~se~~calculating it. If we removed this, then processes could simultaneously be in the critical section as number[j] may be 0 ~~before~~ when process i checks.

B. Suppose there are 14 processes i.e. Process 0 – 14. Some processes require the critical section. The order of execution in which they require critical section is:

P0,P3,P4,P1,P2,P3,P4,P8,P9,P8,P11,P6

Fill the status of the circular Queue after each process requests entry into critical section, i.e. after execution of line 4 in the above code: **[6 marks]**

Process #	Queue
P0	(1,0,0,0,0,0,0,0,0,0,0,0,0,0)
P3	(1,0,0,2,0,0,0,0,0,0,0,0,0,2)
P4	(1,0,0,0,3,0,0,0,0,0,0,0,0,2)
P1	(1,4,0,2,3,0,0,0,0,0,0,0,0,2)
P2	(1,4,5,2,3,0,0,0,0,0,0,0,0,2)
P3	(1,4,5,6,3,0,0,0,0,0,0,0,0,2)
P4	(1,4,5,6,7,0,0,0,0,0,0,0,0,2)
P8	(1,4,5,6,7,0,0,0,8,0,0,0,0,2)
P9	(1,4,5,6,7,0,0,0,8,9,0,0,0,2)
P8	(1,4,5,6,7,0,0,0,10,9,0,0,0,2)
P11	(1,4,5,6,7,0,0,0,10,9,0,11,0,2)
P6	(1,4,5,6,7,0,12,0,10,9,0,11,0,2)

C. At some point, the process numbers and their corresponding ticket numbers are shown.

Process #:	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Ticket#:	**7**	**0**	**0**	**5**	**0**	**3**	**0**	**0**	**0**	**6**	**0**	**3**	**0**	**0**

What is the meaning of Ticket# = 0? **[1 mark]**

That the process does not want access to critical section. (1)

D. Considering the information contained in Part C, replace the blank with a proper process number: **[5 marks]**

Process # 0 waits for process # 9 ✓
Process # 3 waits for process # 5 ×
Process # 5 waits for process # ~~None~~ 11 ×
Process # 9 waits for process # 3 ✓
Process # 11 waits for process # ~~None~~ ×

(2)

Question 6: (8 Marks)

[Write your answer on the Question Paper only in the provided space below]

Given the code segment provided, which creates multiple child processes, assume that the operating system maintains a variable NEXT_PROCESS_ID initialized to 100. Each time a new process is created, it is assigned the value of NEXT_PROCESS_ID as its process ID. The NEXT_PROCESS_ID is then incremented to prepare for the next process creation request. There are no compilation or execution errors in this code.

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

#define BUFFER_SIZE 25
```

```
int main () {

  char msg[BUFFER_SIZE] = "Welcome";
  pid_t pid = fork ();

  if (pid > 0) {
  strcpy (msg, "Welcome to OS course");
  printf ("Parent process waiting for child termination \n"); wait (NULL);
  printf ("Parent Terminating \n"); }
  else {
  printf ("Message: %s \n", msg);
  pid_t pid1 = fork ();

   strcpy (msg, "OS course");
   pid_t pid2 = fork ();

   if (pid2 == 0) {
          strcpy (msg, "Adv OS course");
          printf ("Child Process called \n");
   }
   else {
           wait (NULL);
           printf ("Message: %s \n", msg);
   }
    if(pid1 > 0) {
          wait(NULL);
    }
   }
   return 0;
  }
  return 0;
```

Child Process called
Message: OS course
Message: OS course
Child Process called
Parent Terminating

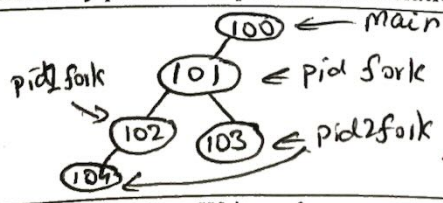
child
2 created
2 outputs
4 created
101, 102
103

ignoring these because it IS a compilation error.

a) How many *new* processes are created *(do not count the initial main() process)*? **[2 Marks]**

4

b) Create a **process tree** illustrating each process node along with its corresponding process ID. The tree should distinctly present the parent-child relationships for all processes. **[3 Marks]**



c) Show output of the code below. Write only one sequence if you feel that multiple sequences can be printed. **[3 Marks]**

~~Welcome~~ Parent process waiting for child termination
Message: Welcome

Child Process called
Message: OS Course
Message: OS Course
Child Process called
Parent Terminating

8

Question 7: (30 Marks)
[Write your answer on the Question Paper only in the provided space below]

Suppose that four processes, P1, P2, P3 and P4, running simultaneously in a multiprocessor system consisting of 3 **processors**. Following is the data about the processes and scheduling policies.

- P1 has three threads. P2 has four threads. P3 has five threads. P4 has three threads.
- All threads in all the processes are CPU-bound, i.e., they never block for I/O.
- There is a strong processor affinity and no load balancing policy in place. i.e. once a thread is tied to a processor, it completes its execution with the same processor.
- There is kernel level thread scheduling in place.
- Queue 1 serves Processor 1, Queue 2 serves Processor 2 and Queue 3 serves Processor 3.
- All the processors are running Round Robin scheduling. Following are the time quantum for each queue.
  - Queue 1 time quantum (q) = 1 ms
  - Queue 2 time quantum (q) = 3 ms
  - Queue 3 time quantum (q) = 2 ms
- All the threads have been loaded in their respective Queues and the current status of the queues at time interval 10 is as shown below (P1T1 means Process 1 – Thread 1 and so on):

Queue 1

P1T1	P3T2	P4T2	P3T5	P2T4	Processor 1

Queue 2

P3T1	P4T1	P1T3	P2T1	Processor 2

Queue 3

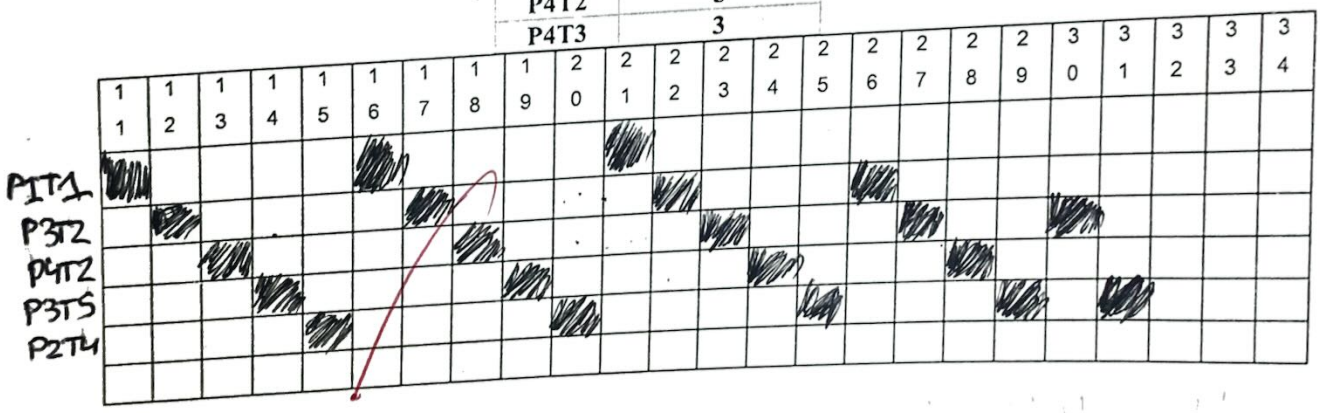
P4T3	P3T3	P2T2	P2T3	P1T2	P3T4	Processor 3

There is currently no process in the processor at time interval **11** i.e. the last process in each of the processor has been context switched out and dispatcher can schedule the next thread immediately at time **11**. The context switch (dispatch latency) cost is not given, so assume it as zero.

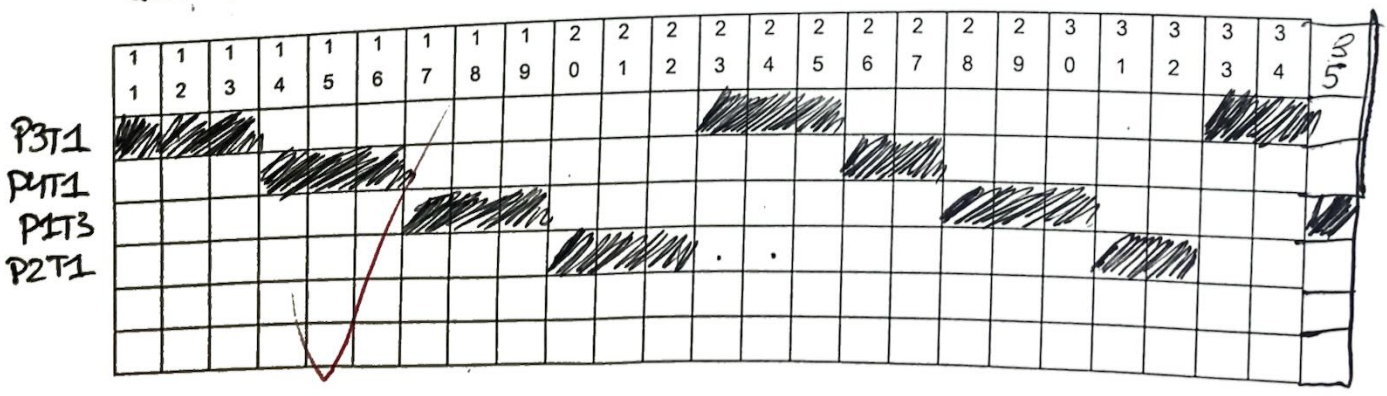
Fill in the following Gantt charts below for each Processor to answer the questions given at the end:

Threads	Remaining CPU Time (ms)
P1T1	3
P1T2	5
P1T3	7
P2T1	5
P2T2	4
P2T3	4
P2T4	5
P3T1	8
P3T2	4
P3T3	5
P3T4	3
P3T5	4
P4T1	5
P4T2	5
P4T3	3

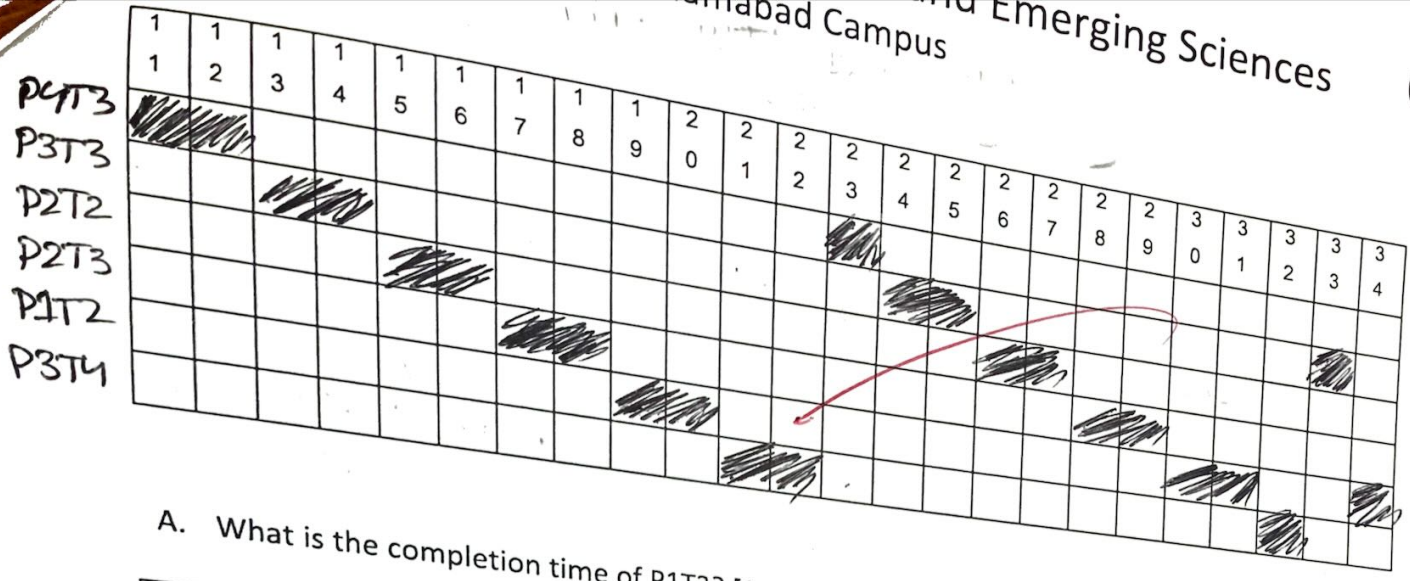
Queue 1 / Processor 1 – Gantt Chart **[3 marks]**



Queue 2 / Processor 2 – Gantt Chart **[3 marks]**



Queue 3 / Processor 3 – Gantt Chart **[3 marks]**



A. What is the completion time of P1T3? **[1 mark]**

T= 35

B. What is the completion time of P1T2? **[1 mark]**

T=34

C. What is the completion time of P2T4? **[1 mark]**

T=31

D. What is the completion time of P3T2? **[1 mark]**

T=26

E. What is the completion time of P3T3? **[1 mark]**

T=33

F. What is the completion time of P4T2? **[1 mark]**

T=30

G. What is the average waiting time of Process P1 after interval 10 (i.e. starting from 11) ? **[2 marks]**

Avg waiting time of P1=(waiting time of P1+w.t of P2+w.t of P3)/3=(8+18+19)/3=45/3=15

H. What is the average waiting time of Process P2 after interval 10 (i.e. starting from 11) ? **[2 marks]**

= (16+17+13+15)/4 = 15.25

I. What is the average waiting time of Process P3 after interval 10 (i.e. starting from 11) ? **[2 marks]**

=(12+14+16+18+19)/5 = 15.8

J. What is the average waiting time of Process P4 after interval 10 (i.e. starting from 11) ? **[2 marks]**

=(15+12+10)=37/3= 12.3

K. What percentage of all processor's time will be spent running Process P1 during interval 11-19? [1 mark]

Proc.1 out of 9 spends 2 quantums, Proc.2 spends 3 out of 9, Proc.3 spends 1 of 9. Total = $\frac{2+3+1}{27}$ = 22.2%.

L. What percentage of all processor's time will be spent running Process P1 during interval 15-24? [1 mark]

Proc.1: 2 of 10 intervals, Proc2: 3 of 10, Proc.3: 2 of 10. total = $\frac{2+3+2}{30}$ = 23.3%.

M. What percentage of all processor's time will be spent running Process P2 during interval 15-24? [1 mark]

Proc.1: $\frac{2}{10}$, Proc.2: $\frac{3}{10}$, Proc.3: $\frac{4}{10}$ $(\frac{2+2}{10})$ => $\frac{2+3+4}{30}$ = 30%.

N. What percentage of all processor's time will be spent running Process P3 during interval 15-24? [1 mark]

Proc1: $\frac{4}{10}$, Proc2: $\frac{2}{10}$, Proc3 $\frac{3}{10}$ => $\frac{4+2+3}{30}$ = $\frac{9}{30}$ = 30%.

O. What percentage of all processor's time will be spent running Process P3 during interval 20-29? [1 mark]

Proc 1: $\frac{4}{10}$, Proc2: $\frac{3}{10}$, Proc 3: $\frac{4}{10}$ => $\frac{4+3+4}{30}$ = $\frac{11}{30}$ = 36.6%

P. What percentage of all processor's time will be spent running Process P4 during interval 20-29? [1 mark]

Proc1: $\frac{2}{10}$, Proc 2: $\frac{2}{10}$, Proc 3: $\frac{1}{10}$ => $\frac{2+2+1}{30}$ = $\frac{5}{30}$ = 16.6%.