

PAPER CODE	EXAMINER	DEPARTMENT	TEL
CPT104	XXXXXX	XXXXXXXXX	XXXXXX

2021/22 SEMESTER 2 – Resit

Open Book Exam

BACHELOR DEGREE – Year 2

Operating Systems Concepts

TIME ALLOWED: 2 hours

INSTRUCTIONS TO CANDIDATES

- 1. Total marks available are 100, accounting for 100% of the overall module marks.**
- 2. Answer all FOUR questions.**
- 3. The number in the column on the right indicates the marks for each question.**
- 4. Relevant and clear steps should be included in the answers.**
- 5. The university approved calculator - Casio FS82ES/83ES can be used.**
- 6. All the answers must be in English in the answer script provided.**

QUESTION I. Fundamentals**(45 marks)**

1. What is the difference between a **page** and a **frame**? (6 marks)

2. Why is it important to minimize the **context switch** time in **Real-Time scheduling**? (6 marks)

3. Describe how you would convince a coworker to better manage a personal computer by performing regular backups and keep the system patches current. Briefly discuss **three important methods**. (12 marks)

4. Consider the following program segments for two different processes (P1, P2) executing **concurrently** where **a** and **b** are not shared variables, but **x** starts at **0** and is a shared variable.

Processor #1

for (a = 1; a <= 3; a++)

x = x + 1;

Processor #2

for (b = 1; b <= 3; b++)

x = x + 1;

If the processes P1 and P2 execute only once at any speed, what are the possible resulting values of **x**? Explain your answers. (4 marks)

5. Give **two** arguments for the use of **Virtual Machines**, and **two** arguments against it. Explain your answers. (12 marks)

6. Why is switching **threads** less costly than switching **processes**?

(5 marks)

QUESTION II. CPU scheduling, Memory management, Disk scheduling

(31 marks)

1. Calculate the number of page faults for the following sequence of page references (each element in the sequence represents a page number) using the **First-In, First-Out FIFO algorithm** with frame size of **3**.

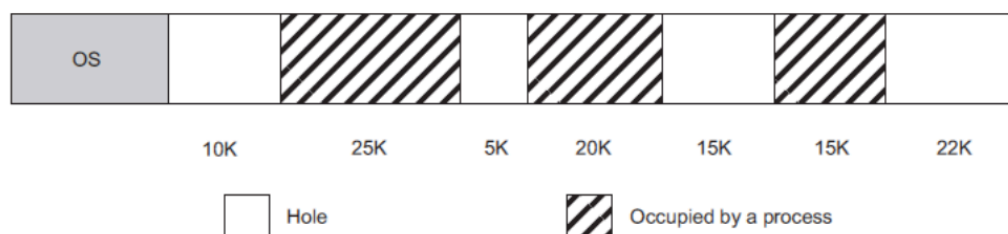
0 1 7 0 1 2 0 1 2 3 2 7 1 0 3 1 0 3

(6 marks)

2. Consider the memory allocation scenario as shown in the figure below.

Allocate memory for additional requests of 4K and 10K (in this order).

Compare the memory allocation, using **first-fit**, **best-fit**, and **worst-fit allocation** methods, in terms of internal fragmentation. (12 marks)



3. Consider the following scenario of processes with their priority:

Process	Arrival time	Execution time	Priority number
P1	0	3	3
P2	2	7	4
P3	3	5	1
P4	5	9	2

Draw the Gantt chart for the execution of the processes using the **Priority scheduling**. (2 marks)

Calculate waiting time for each process and average waiting time for the system. (5 marks)

4. Consider a disk queue with I/O requests on the following cylinders in their arriving order:

17, 30, 24, 37, 15, 27, 11, 75, 20, 5

We assume a disk with **80** tracks and the disk head is initially located at track **28**.

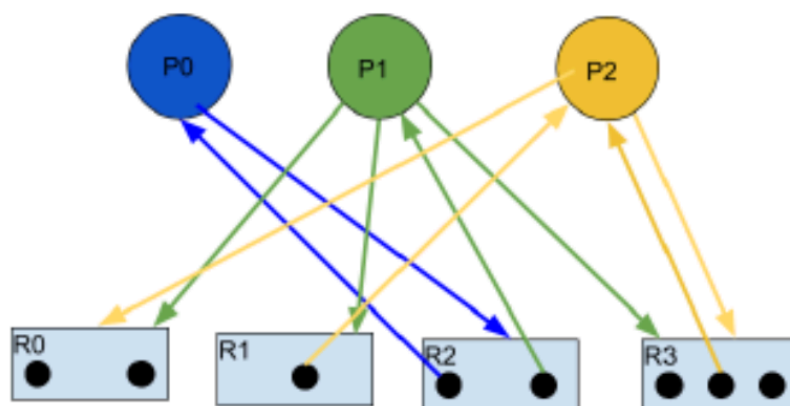
Write the sequence in which requested tracks are serviced using the **Shortest Seek Time First (SSTF) algorithm** and calculate the **total head movement** (in number of cylinders) incurred while servicing these requests. (6 marks)

QUESTION III. Resource allocation**(12 marks)**

A system is having 3 processes: P1, P2 and P3 and 4 resource types:

R0 (2 instances), R1 (1 instance), R2 (2 instances) and R3 (3 instances).

The **Resource Allocation Graph** (RAG) of the system is shown below:



Convert the RAG to the matrix representation (**Allocation**, **Need** and **Available**). (7 marks)

Determine if there is a **deadlock**. If yes, indicate the processes and resources involved.

(5 marks)

QUESTION IV. Operating System in C Language**(12 marks)**

Consider the **Producer-Consumer** synchronization problem.

The shared buffer size is N .

Three semaphores *empty*, *full* and *mutex* are defined with respective initial values of N , 0 and 1, where *empty* denotes the number of available buffers for the producer to fill in, *full* denotes the number of available buffers for consumer to pick up, and *mutex* provides mutual exclusion to access the buffer pool.

The valid semaphore operations are: *wait* () and *signal* ().

Producer:	Consumer:
1. <code>do {</code>	1. <code>do {</code>
2. <code>wait (empty) ;</code>	2. <code>wait (full) ;</code>
3. <code>wait (mutex) ;</code>	3. <code>wait (mutex) ;</code>
4. <code>//Add item to buffer</code>	4. <code>//Consume item from buffer</code>
5. <code>signal (mutex) ;</code>	5. <code>signal (mutex) ;</code>
6. <code>signal (full) ;</code>	6. <code>signal (empty) ;</code>
7. <code>} while (1) ;</code>	7. <code>} while (1) ;</code>

1. What happens if we remove the semaphore *full* from both producer and consumer (i.e. line 6 in producer and line 2 in consumer are deleted)? (2 marks)
2. What happens if the semaphore *empty* and *mutex* is placed in reverse order in producer (i.e. line 2 and line 3 in producer are swapped)? (5 marks)
3. What happens if line 6 in the producer is swapped with line 6 in the consumer? (i.e. *signal* (full) and *signal*(empty) are swapped) (5 marks)

END OF EXAM PAPER