Xi’an Jiaotong-Liverpool University

PAPER CODE	EXAMINER	DEPARTMENT	TEL
CPT104	Gabriela Mogos Dongyao Jia	Department of Computing	1515

2022/23 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 (40 marks)

1. Round-Robin schedulers use a fixed size time quantum for allocating CPU time.

Large time quantum sizes provide certain advantages to the system, while small time quantum sizes provide other advantages.

Assume that you are designing a system where throughput is more important than response time, while use of Round-Robin scheduling is required. Explain whether you would use a relatively large or relatively small quantum value for such a system, and why.

(6 marks)

2. Let's consider a process executing on a CPU.

Give **3 reasons** for this process to be blocked, and so to be preempted from the CPU by the Operating System? (6 marks)

What are the conditions for that process to be granted again the CPU? (6 marks)

3. In a system, there are three processes, **P1**, **P2**, and **P3**, divided into 32, 189, and 65 pages, respectively.

If there are 115 frames in the **memory**, then calculate the proportions in which the frames will be allocated to the processes. (6 marks)

4. In a Local Area Network LAN environment, some users are working on a project.

The project leader, with his four members working on the project, wants that all group members should be able to **read** and **write** on the project directory, but should not be able to **delete** it.

Other users may be allowed only to read and execute the files under the project directory.

Besides this, the project leader should have all the access rights.

Prepare a file access protection scheme for this.

(6 marks)

5. Describe **five** major activities of an **operating system** with regard to process management?

(10 marks)

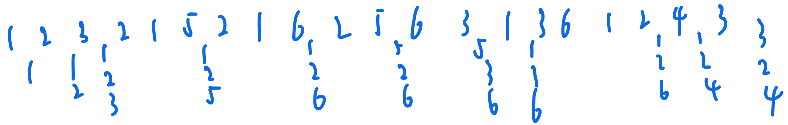
QUESTION II. CPU scheduling, Memory management, Disk scheduling

(36 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 **Least Recently Used (LRU) algorithm** with frame size of **3.**

1 2 3 2 1 5 2 1 6 2 5 6 3 1 3 6 1 2 4 3

(6 marks)



11

2. In a paging system with **Translation Lookaside Buffer** (TLB), it takes 40 ns to search the TLB and 100 ns to access the memory. If the TLB hit ratio is 60%, find the effective memory access time.
What should be the **hit ratio** to achieve the effective memory access time of 170 ns? (6 marks)

$60\%(100+40)+(1-60\%)(100\times2+40)=180\ ns$

$a\cdot140+(1-a)240=170$

$70=100a$

$a=70\%$

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

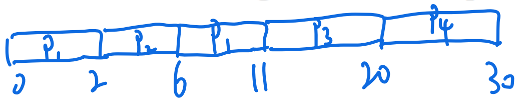
Process	Arrival time	Burst time
P1	0	7
P2	2	4
P3	3	9
P4	5	10

Explain the **Shortest-Remaining-Time-First (SRTF)** algorithm of the processes. (2 marks)

Draw the Gantt chart for the execution of the processes. (2 marks)

Calculate the **Waiting Time** for each process and the **Average Waiting Time** for the system.

(6 marks)



$P_1: 11-0-7=4$

$P_2: 6-2-4=0$

$P_3: 20-3-9=8$

$P_4: 30-5-10=15$

Average : $\frac{4+0+8+15}{4}=6.75$

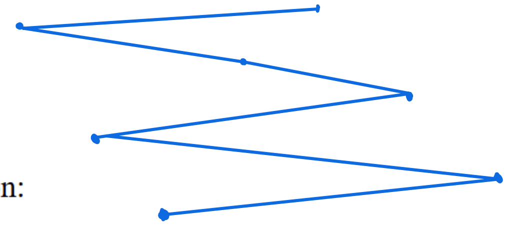
4. Consider a disk queue with I/O requests on the following cylinders in their arriving order:

25, 90, 135, 50, 190, 60

We assume a disk with **200** tracks and the disk head is initially located at track **100**.

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

25 50 60 90 100 135 190



100-25 + 135-25 + 135-50

+ 190-50 + 190-60

= 540

5. Given the following information:

Job List:		**Memory Block List:**	
Job Number	**Memory Requested**	**Memory Block**	**Memory Block Size**
Process A	57K	Block 1	900K
Process B	920K	Block 2	910K
Process C	50K	Block 3	200K
Process D	701K	Block 4	300K

Use the **Best-fit algorithm** to indicate which memory blocks are allocated to each of the arriving processes. Explain. (8 marks)

Process A: 57k, 200k is the smallest ≥57k, so Block 3 is allocated

B 920k, No Block

C 50 k, 300k is the smallest ≥50k, So Block 4 is allocated

D 701 k, 900k is the smallest ≥701k, so Block 1 is allocated

QUESTION III. Resource allocation (12 marks)

Consider a system with the following information.

Available resources

R1	R2	R3	R4
2	1	0	0

Process	Max				Allocation			
	R1	R2	R3	R4	R1	R2	R3	R4
P1	0	0	1	2	0	0	1	2
P2	2	7	5	0	2	0	0	0
P3	6	6	5	6	0	0	3	4
P4	4	3	5	6	2	3	5	4
P5	0	6	5	2	0	3	3	2

Need

R_1	R_2	R_3	R_4
0	0	0	0
0	7	5	0
6	6	2	2
2	0	0	2
0	3	2	0

Is this system currently in a safe or unsafe state? Why? Explain. $P_1 \rightarrow P_4 \rightarrow P_5 \rightarrow P_2 \rightarrow P_3$ (6 marks)

If a request from P3 arrives for (0, 1, 0, 0), can that request be safely granted immediately? Explain the answer. (6 marks)

$(0,1,0,0) \le (6,6,2,2)$ $(0,1,0,0) < (2,1,0,0)$

P_2 and P_3 fail, so the request cannot be safely granted immediately.

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

The following is a C language program using **POSIX** pipes:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main()
{
    int fd[2];
    pid_t pid;

    if (pipe(fd) == -1) {
        perror("pipe");
        exit(EXIT_FAILURE);
        }

        pid = fork();
        if (pid == -1) {
                perror("fork");
                exit(EXIT_FAILURE);
        }

        if (pid == 0) {
                close(fd[1]);
                char buffer[256];
                int nread = read(fd[0], buffer, sizeof(buffer));
                printf("child: read %d bytes from the pipe: %s\n", nread, buffer);
                close(fd[0]);
                exit(EXIT_SUCCESS);
        } else {
                close(fd[0]);
                char message[] = "Hello, pipe!";
                write(fd[1], message, sizeof(message));
                printf("parent: wrote message to the pipe\n");
                close(fd[1]);
                exit(EXIT_SUCCESS);
        }
}
```

(1) What is the program as a whole attempting to do? (4 marks)

The parent process and child process use an ordinary pipe to communicate. The parent writes the message and the child reads.

(2) What is the output of this program? (4 marks)

13 空格!

parent: wrote messge to the pipe   child: read ~~12~~ bytes from the pipe: Hello, pipe!

(3) How to improve the above program to support the child process sending messages to the parent process at the same time? Please describe in text. (4 marks)

Create a second pipe.

END OF EXAM PAPER