The Open University of Sri Lanka
Faculty of Engineering Technology
Department of Electrical and Computer Engineering

Study Programme	: Bachelor of Technology Honours in Engineering
Name of the Examination	: Final Examination
Course Code and Title	**: EEX5535/ECX5235 Operating Systems**
Academic Year	: 2019/2020
Date	: 12th of August 2020
Time	: 1330-1630hrs
Duration	: **3 hours**

General Instructions

1. Read all instructions carefully before answering the questions.
2. This question paper consists of **Six (6)** questions in **Four (4)** pages.
3. Answer **any five (5)** questions given. All questions carry equal marks.
5. Answer for each question should commence from a new page.
6. This is a Closed Book Test **(CBT).**
7. Answers should be in clear hand writing.
8. Do not use red colour pen.

Question 1

a) Briefly explain how the distinction between kernel mode and user mode functions as an elementary form of protection (security) system.

[04 Marks]

b) Some early computers protected the operating system by placing it in a memory partition that could not be modified by either the user job or the operating system itself. Describe two difficulties that you think could arise with such a scheme.

[04 Marks]

c) Answer the following questions regarding the cache.

   a. State two reasons why caches are useful.

   [04 Marks]

   b. What problems do they cause?

   [04 Marks]

   c. If a cache can be made as large as the device for which it is caching (for instance, a cache as large as a disk), why not make it that large and eliminate the device?

   [04 Marks]

Question 2

Answer the following questions on memory management.

a) Briefly explain the effect of allowing two entries in a page table to point to the same page frame in memory.

[02 Marks]

b) Explain how this effect could be used to decrease the amount of time needed to copy a large amount of memory from one place to another.

[02 Marks]

c) What effect would updating some bytes on the one page have on the other page?

[04 Marks]

d) Given six memory partitions of 300 KB, 600 KB, 350 KB, 200 KB, 750 KB, and 125 KB (in order).

   a. How would the *first-fit, best-fit,* and *worst-fit* algorithms place processes of size 115 KB, 500 KB, 358 KB, 200 KB, and 375 KB (in order)?

   [06 Marks]

   a. Calculate internal/external fragmentation for each algorithm.

   [04 Marks]

   b. Rank the algorithms in terms of how efficiently they use memory.

   [02 Marks]

Question 3

Assume that there are 5 processes, P0 through P4, and 4 types of resources.
The number of available instances of the resources A, B, C and D at T0 for a particular time instance is given below.

Available			
A	B	C	D
1	5	2	0

At T0, we have the following system state:

	Allocation				Max			
	A	B	C	D	A	B	C	D
P0	0	1	1	0	0	2	1	0
P1	1	2	3	1	1	6	5	2
P2	1	3	6	5	2	3	6	6
P3	0	6	3	2	0	6	5	2
P4	0	0	1	4	0	6	5	6

a) Create the need matrix

[02 Marks]

b) Use the safety algorithm to test if the system is in a safe state. Show the steps of how the safety sequence is found.

[10 Marks]

c) If the system is in a safe state, can the following requests be granted, why or why not? Run the safety algorithm on each request as necessary.
   a. P1 requests (2,1,1,0)
   b. P1 requests (0,2,1,0)

[08 Marks]

Question 4

Consider the following set of processes, with the arrival time and the length of the CPU burst time given in seconds.

Process	Arrival Time (hh:mm:ss)	CPU Burst Time (Seconds)
P 0	00:00:00	2
P 1	00:00:01	2
P 2	00:00:01	1
P 3	00:00:02	3
P 4	00:00:09	2
P 5	00:00:11	3
P 6	00:00:13	5

a) Draw four (4) Gantt charts that illustrate the execution of these processes using First Come First Served (FCFS), Shortest Job First (SJF), Shortest Remaining Time First(SRTF) and Round Robin(RR) algorithms. Choose proper quantum size for RR scheduling and clearly state the reasons to select the quantum size. State assumptions clearly if any.

[10 Marks]

b) At each scheduling method calculate;
   a. the average turnaround time
   b. the average waiting time
   c. throughput

[06 Marks]

c) Which one of the above four algorithms is appropriate for this system? Justify your answer.

[04 Marks]

Question 5

a) Briefly describe the actions taken by a kernel to context switch between processes.

[05 Marks]

b) Differentiate short-term scheduling from medium-term and long-term scheduling.

[06 Marks]

c) Identify the values of *pid* at lines A,B,C and D in the given program. Assume that the actual *pids* of the parent and child are 1200 and 1203 respectively. State all your assumptions if any. Briefly explain the basis for your answer.

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

int main ()
{
    /* fork a child process */
    pid = fork();
    if ( pid < 0 ) { /*error occurred
        fprintf(stderr, "Fork Failed");
        return 1;
    }
    else if (pid ==0) { /* child process
        pid1 = getpid();
        printf("child: pid = %d" ,pid); /
        printf("child: pid1 = %d" ,pid1);
    }
    else { /* parent process */
        pid1 = getpid();
        printf("parent: pid = %d" ,pid);
```

[09 Marks]

Question 6

a) Most modern operating systems provide features enabling a process to contain multiple threads of control.

i. Illustrate the differences between a traditional single-threaded process and a multithreaded process using a suitable diagram. Label the basic units comprised in a thread.

[04 Marks]

ii. Explain the sections of a thread shared between other threads belonging to the same process.

[02 Marks]

b) Discuss the benefits of multithreaded programming under the given four categories. Give one example for each benefit.

i. Responsiveness
ii. Resource sharing
iii. Economy
iv. Scalability

[08 Marks]

c) The program shown below uses the Pthreads API. What would be the output from the program at *Line C* and *Line P*? Briefly explain the basis for your answer.

[06 Marks]

```
#include <pthread.h>
#include <stdio.h>

int value = 0;
void *runner(void *param);  /* the thread */


int main(int argc, char *argv[])
{
    int pid;
    pthread_t tid;
    pthread_attr_t attr;

    pid = fork();

    if (pid == 0) {  /* child process */
        pthread_attr_init(&attr);
        pthread_create(&tid, &attr, runner, NULL);
        pthread_join(tid, NULL);
        printf("CHILD: value = %d\n", value);  /*
    }
    else if (pid > 0) {  /* parent process */
        wait(NULL);
        printf("PARENT: value = %d\n", value);  ,
    }
```

---------- **End of the paper** ----------