United International University (UIU)

Dept. of Computer Science & Engineering (CSE)

***Final Exam*: Trimester: Fall 2019**

Course: CSI 309, Operating System Concepts

Marks: 40, Time: 2 hours

Answer all the questions. Numbers to the right of the questions denote their marks.

1. a) Will **Peterson's solution** work properly when two processes call ***enter_region()*** at the same time? Justify your answer. Next, consider the following code snippet for philosopher, *i*:

```
void put_forks(i)
{
    down(&mutex);
    state[i] = THINKING
    test(LEFT);
    test(RIGHT);
    up(&mutex);
}
```

What are the purposes of **mutex**, **test(LEFT)**, and **test(RIGHT)** in this code? [2+3]

b) Consider a system executing **four processes – A, B, C, and D**. A has two instructions - *A1* and *A2*, B has two instructions - *B1* and *B2*, C has *C1*, and D has *D1*. All of these instructions are inside a loop in their respective processes. For example, consider process A:

```
while (TRUE) {
    A1;
    A2; }
```

Now write an algorithm to maintain synchronization among these processes as follows: [4]

A1 --> B1 --> D1 --> C1 --> A2 --> D1 --> A1 . . . . .

Also, show a (any) scenario to explain how the wrong ordering of using semaphores can lead to a potential deadlock. [1]

c) Write down the disadvantage(s) of **linked list file allocation technique.** Assume, there are 12 blocks (0-11) in a secondary storage disk. Blocks 1, 2, 4, 7, and 8 have already been allocated to different files. Now, allocate blocks for file F1 (size = 2.5 blocks) and then file F2 (size = 2 blocks) using **(i) contiguous file allocation** and **(ii) linked list allocation using a table in memory**. Note that you have to allocate F1 and F2 sequentially. [1+2+2]

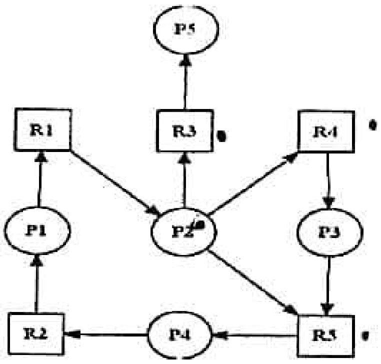
2. a) Round Robin algorithm takes new scheduling decision after each time quantum. Now consider the following processes with arrival time and burst time. Draw the **Gantt chart** and calculate the **average turnaround time** using **Round Robin** with **time quantum** 4? [5]

Process	*Arrival Time*	*Burst Time*
P1	1	5
P2	0	6
P3	3	7
P4	1	9
P5	1	2
P6	7	3

b) Discuss different **(four)** possible implementations of a **Web server** with their advantages and disadvantages. [5]

c) Which information are shared by different threads of same process? What will happen if a multitasking (taking input, saving in disk, displaying, etc.) word processor is implemented with a single thread? Differentiate between User-level and Kernel-level threading. [1+2+2]

3. a)



Detect deadlock (if any) in the above scenario using deadlock detection algorithm for single resource of each type. You need to **run the algorithm three times**; considering **R3, P3,** and **P2** as the initial nodes. You need to show the list at least whenever any backtracking is needed. Also, you must use alphabetical ordering for selecting an unmarked arc to traverse. For example- from P2, traverse through R3 first, then R4, and finally R5. [4]

b) Suppose there are **3 processes - P1, P2, P3**; and **3 resources - R1, R2, R3.** Now consider the following sequence of statements: [2]

i. P1 requests R3
ii. P1 requests R2
iii. P2 requests R2

iv. P3 requests R3
v. P2 requests R1
vi. P1 releases R3

Now draw the resource allocation graph after each of the above statements have been executed.

c) Suppose there are **three processes (A, B, and C)** in the system. Existing and allocated resource lists, current allocation, and request matrix are provided as follows. Run a **multiple resource deadlock detection algorithm** over the following example, and decide which processes will be in deadlock.

[Once a process gets all of its requirements, it releases them.] [4]

$$\begin{array}{ccccc} & \text{Tape drives} & \text{Plotters} & \text{Scanners} & \text{CD Roms} \\ E = (& 4 & 2 & 3 & 1\ ) \end{array} \qquad \begin{array}{ccccc} & \text{Tape drives} & \text{Plotters} & \text{Scanners} & \text{CD Roms} \\ A = (& 2 & 1 & 0 & 0\ ) \end{array}$$

Current allocation matrix

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \\ 0 & 1 & 2 & 0 \end{bmatrix}$$

Request matrix

$$R = \begin{bmatrix} 2 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$