**Department of Virtualization**
**School of Computer Science**

|  |  |
|---|---|
| **Assignment No.** 1 | **Deadline:** 22/11/18 |
| **Subject:** Operating Systems | **Code: CSEG 2007** |
| **Program/Branch:** B.Tech (CSE) – CCVT | **Semester:** III |
| **Faculty:** Dr. Amit Agarwal | **Max. Marks: 20** |

**Q1**. Suppose that the following processes arrive for execution at the times indicated. Each process will run the listed amount of time. In answering the questions, use non-preemptive scheduling and base all decisions on the information you have at the time the decision must be made.

| Process | Arrival Time | Burst Time |
|---|---|---|
| P1 | 0.0 | 8 |
| P2 | 0.4 | 4 |
| P3 | 1.0 | 1 |

a. What is the average turnaround time for these processes with the FCFS scheduling algorithm?

b. What is the average turnaround time for these processes with the SJF scheduling algorithm?

c. The SJF algorithm is supposed to improve performance, but notice that we chose to run process P1 at time 0 because we did not know that two shorter processes would arrive soon. Compute what the average turnaround time will be if the CPU is left idle for the first 1 unit and then SJF scheduling is used. Remember that processes P1 and P2 are waiting during this idle time, so their waiting time may increase. This algorithm could be known as future-knowledge scheduling.

**Q2.** In a system, the following state of processes and resources are given:
R2 → P1, P1 →R2, P2 → R3, R1 → P2, R3 → P3, P3 → R4, P4 → R3, R4 → P4, P4 → R1, R1 → P5. Draw the Resource allocation graph and wait for graph for the system and check the deadlock condition.

**Q3.**A shared variable x, initialized to zero, is operated on by four concurrent processes W, X, Y, Z as follows. Each of the processes W and X reads x from memory, increments by one, stores it to memory, and then terminates. Each of the processes Y and Z reads x from memory, decrements by two, stores it to memory, and then terminates. Each process before reading x invokes the P operation (i.e., wait) on a counting semaphore S and invokes the V operation (i.e., signal) on the semaphore S after storing x to memory. Semaphore S is initialized to two. What is the maximum possible value of x after all processes complete execution?

**Q4.**Including the initial parent process, how many processes are created by the following program?

```
#include <stdio.h>
#include <unistd.h>
int main()
{
fork();fork();fork();fork();
return 0;
}
```

**Q5**. Consider a system with the following information. Determine whether the system is in safe state or not.

**Total Resource**

| R1 | R2 | R3 |
|---|---|---|
| 15 | 8 | 8 |

| **Process** | **Max** | | | **Alloc** | | |
|---|---|---|---|---|---|---|
| | **R1** | **R2** | **R3** | **R1** | **R2** | **R3** |
| **P1** | 5 | 6 | 3 | 2 | 1 | 0 |
| **P2** | 8 | 5 | 6 | 3 | 2 | 3 |
| **P3** | 4 | 6 | 2 | 3 | 0 | 2 |
| **P4** | 7 | 4 | 3 | 3 | 2 | 0 |
| **P5** | 4 | 3 | 3 | 1 | 0 | 1 |

**Q6.** Consider the following processes as shown in Table:

| **Process** | **Arrival Time** | **CPU Burst Time** | **Priority** |
|---|---|---|---|
| P1 | 0 | 8 | 1 |
| P2 | 1 | 20 | 3 |
| P3 | 2 | 3 | 2 |
| P4 | 3 | 6 | 5 |
| P5 | 4 | 12 | 4 |

Draw the Gantt chart for the execution of the processes, showing their start time and end time using FCFS, SJF (Non pre-emptive), Priority based scheduling (Pre-emptive) and Round Robin (Time Quantum = 5). Compute the *average waiting time* and *average turnaround time* for each scheduling algorithm.

**Q7.** Consider variable size partitioning technique, where holes available are 20MB, 30MB, 44MB and 12MB of size. A Process of size 19MB is allocated as per First Fit, Best Fit and Worse Fit algorithms. Then write in percentage the space wasted in each case.

**Q8.** Consider 100 Hz CPU, three processes which require 10, 20 and 30 secs and arrive at times 0, 2 and 6 secs respectively. How many context switches are needed if the operating system implements a shortest remaining time first (SRTF) scheduling algorithm? Do not count the context switches at time zero and at the end. Consider cycles elapsed for one context switch is 10 cycles. Compute percentage of time spent in context switching.

**Q9.** A system has 6 identical resources and *n* processes competing for them. Each process can request atmost 2 resources. Which value of *n* could lead to a deadlock?

**Q10.** You have the following code. Check whether deadlock can occur in this scenario. If yes then show how it is happening? Also modify this code to remove the situation of deadlock. Also tell which condition you violated to reach your solution. (L1 ,L2 are resources)

```
P1                          P2
acquire(L1) // line 1     acquire(L2) //  line 3
acquire(L2) //line 2       acquire(L1)  //line 4
release(L2)                  release(L1)
release(L1)                  release(L2)
```