Assignment II
Total Marks: 60
Write appropriate programs and/or answer to solve the following problems. • Submit all programs and question answers in a word file • For program submit screenshots of output • Word file name must be CSE325_<section-number>_A2_<your-student-id> • Submission Deadline: 12 April 2021 11:59 AM

1. You know the core concepts behind process management of operating system. OS handles creation of processes, manage different queues such as job, ready and I/O queues. It also uses different process scheduling algorithms to execute processes by maintaining proper scheduling schemes. Now your task is to build a process simulator Your simulator will have following menu page:

Create Process
Ready Process
Show Job Queue
Show Ready Queue
Execute Process
Re-Execute Process
Exit

Your simulator will receive a command as input and perform the task as per the command. You should follow the instructions stated below:

a) Whenever you create a process, your process will have the following properties:

- processID;
- processName;
- processStatus;
- burst;
- arrivalTime;
- priority;
- processCounter;

b) For job queue, ready queue and device queue you can use advanced data structures like Linked List or Vector

c) Ready Process function transferring processes from Job Queue to Ready Queue. Here once process is ready, shift the process from new to ready state

d) Show ready queue option is used to print ready queue
e) Inside the execute process function you have to select different scheduling algorithm. For each of the scheduling algorithm create appropriate gantt chart. Create submenu inside execute process function.

"CPU Scheduling Algorithm Selection"
(1) First-Come-First-Served (FCFS)
(2) Shortest-Job-First (SJF)
(3) Preemptive SJF (shortest-remaining-time-first)
(4) Priority-based Scheduling
(5) Preemptive Priority-based Scheduling
(6) Round-Robin Scheduling

f) Inside Re-execute process you have to set process status as ready of ready queue and start execution again.
g) A sample code is assigned for your process simulator.

2. The computer science department of EWU runs a programming club to help undergraduate students with their programming assignments. The club has a coordinator and several tutors to assist the students. The waiting area of the center has several chairs. Initially, all the chairs are empty. The coordinator is waiting for the students to arrive. The tutors are either waiting for the coordinator to notify that there are students waiting or they are busy tutoring. The tutoring area is different from the waiting area. A student, while programming for his project, decides to go to a club to get help from a tutor. After arriving at the club, the student sits in an empty chair in the waiting area and waits to be called for tutoring. If no chairs are available, the student will go back to programming and come back to the club later. Once a student arrives, the coordinator queues the student based on the student's priority, and then the coordinator notifies an idle tutor. A tutor, once woken up, finds the student with the highest priority and begins tutoring. A tutor after helping a student, waits for the next student. A student after receiving help from a tutor goes back to programming. The priority of a student is based on the number of times the student has visited the club. A student visiting the club for the first time gets the highest priority. In general, a student visiting for the ith time has a priority higher than the priority of the student visiting for the kth time for any $k > i$. If two students have the same priority, then the student who came first has a higher priority.
   Using POSIX threads, mutex locks, and semaphores implement a solution that synchronizes the activities of the coordinator, tutors, and the students. The total number of students, the number of tutors, the number of chairs, and the number of times a student seeks a tutor's help are passed as command line arguments. Once a student thread takes the required number of help from the tutors, it should terminate. Once all the student threads are terminated, the tutor threads, the coordinator thread, and the main program should be terminated. Your program should work for any number of students, tutors, chairs and help sought. Allocate memory for data structures dynamically based on the input parameter(s)

3. There are three smokers and one agent. Each smoker continuously rolls a cigarette and then smokes it. But to roll and smoke a cigarette, the smoker needs three ingredients: tobacco, paper, and matches. One of the smokers has paper, another has tobacco, and the third has matches. The agent has infinite supply of all three materials. The agent places two of the ingredients on the table. The smoker who has the remaining ingredient then makes and smokes the cigarette, signaling the agent on completion. The agent then puts out another two of the three ingredients and the cycle repeats.

   You must output the progress of the processes. E.g. when the agent places the two ingredients you should print that information. When a smoker gets to roll and smoke a cigarette, output which smoker got the ingredients and so-on. Show the run of your processes for a while until around 10 cigarettes are rolled.

4. A variation of the round-robin scheduler is the **regressive round-robin** scheduler. This scheduler assigns each process a time quantum and a priority. The initial value of a time quantum is 50 milliseconds. However, every time a process has been allocated the CPU and uses its entire time quantum (does not block for I/O), 10 milliseconds is added to its time quantum, and its priority level is boosted. (The time quantum for a process can be increased to a maximum of 100 milliseconds.) When a process blocks before using its entire time quantum, its time quantum is reduced by 5 milliseconds, but its priority remains the same. What type of process (CPU-bound or I/O-bound) does the regressive round-robin scheduler favor? Explain.
5. A program is decomposed into 4 sub-processes. All the sub-processes can execute in parallel. However, some processes can execute only after some other processes can finish execution. This is described by the following precedence constraints:
   - Process 3 can start only after processes 1 and 2 finish execution
   - Process 4 can start only after 2 finish execution.

   Provide a pseudo code with semaphores to accomplish this.