OOP + OS Assignment
CPU Scheduling Algorithm Simulator in C++

Instructions:
1. This assignment covers OOP topics such as class, objects, friend functions, default parameters, constructors, operator overloading, etc. In the case of OS, it covers CPU scheduling algorithms.
2. The assignment is of 20 marks in which 10 marks are for OOP and 10 marks are for OS.
3. OOP concepts can be used upto the topics covered till 15th December 2022.
3. Copying from classmates or the Internet is strictly prohibited.
4. Create a .zip file consisting of code, output files, readme file and submit it in Google Classroom. The readme file should explain briefly your code. The name of the .zip file should be **yourname_enrollment.zip**.

Deadline: 3rd January 2023
Total Marks: 20 (10 for OOP and 10 for OS)

This assignment is a mix of Operating Systems and Object Oriented Programming concepts. In this assignment, the student has to design a simulator to simulate the behavior of a CPU scheduler, i.e., choosing a process from the ready queue based on a scheduling algorithm to execute it by the processor. The simulator has to be implemented using C++. The current knowledge of C++ will be sufficient to do this assignment. The marks will be equally divided in both OS and OOP courses. Those who are not crediting the OS course, can learn the concepts of CPU Scheduling from internet or from CSE students.

The simulator will consists of following classes:
(i) Process: The data members of this class should store process id, arrival time in the ready queue, CPU burst time, completion time, turn around time, waiting time, and response time. The member functions of this class should assign values to the data members and print them. A constructor should also be used.

(ii) Process_Creator: This class will create an array of processes and assign a random arrival time and burst time to each process. Data members, constructor and member functions can be written accordingly.

(iii) Scheduler: This class will implement the scheduling algorithm. The class will maintain a ready queue of infinite capacity (i.e., any number of processes can be accommodated in the queue). The ready queue should be implemented using the min-Heap data structure where the highest priority process will be the root. The priority of the process will depend on the scheduling algorithm.
The ready queue (i.e., min-heap) should be implemented as a class where different heap operations should be its member functions. Students have to implement three scheduling algorithms, viz., First Come First Serve (FCFS), Round Robin and Completely Fair Scheduler (CFS). While running the program, the user should be asked the choice of scheduling algorithm. The CFS scheduling will be implemented using Red Black Tree. So, in case of CFS, ready queue will be implemented using Red Black Tree. To understand the CFS Scheduling following youtube lecture can be followed.
https://www.youtube.com/watch?v=scfDOof9pww

However, other internet sources can also be checked.

(iv) Simulator: This class will start the simulation. If the simulation time is 1 second and arrival time and burst time of a process are in terms of milliseconds then the class will run a ‘for’ loop from

t=0 to 1000. At each iteration, the class will execute all the necessary functions and capture the required values. The class will also print the output in the following ways:

(a) A file named as processes.txt will be created which will contain a table as follows. The file handling coding can be done using C programming.

Process Id	Arrival Time	Burst Time	Completion Time	Turn Around Time	Waiting Time	Response Time

Here, the number of rows will depend on the number of processes.

(b) A file named as status.txt which will print a table. For each millisecond, the file will show the process id of the processes arrived in the system, running by the processor and exiting from the system. The number of rows in the table will depend on the number of times the 'for' loop is running, i.e., the number of milliseconds in the simulation time. However, if in a particular millisecond, no process arrived or exited and the process being run by the processor is the same as that of previous millisecond then the current millisecond can be skipped from printing in the table. For example,

Millisecond Number	Process Id	Status
1	1	Arrived
1	1	Running
2	2	Arrived
5	1	Exit
5	2	Running
5	3	Arrived
8	2	Exit

The main() function should take the following input from the user: (i) Simulation time (ii) Name of scheduling algorithm (iii) Time quantum in case of Round Robin.