1 Scheduling Simulator

Please download the scheduling simulator skeleton code from the moodle. The scheduling simulator illustrates the behaviour of scheduling algorithms against a simulated mix of process loads. The user can specify the number of processes, the mean and standard deviation for compute time and I/O blocking time for each process, and the duration of the simulation. At the end of the simulation, a statistical summary is presented.

We have given the implementation of FIFO scheduling as an example of how to implement a scheduling algorithm in the given environment. You are allowed to use any development environment you prefer. Please go through the instructions carefully and complete the assignment.

2 Running the Simulator

- Compile the java code using the following command.

```
$ javac *.java
```

- The program reads a configuration file (scheduling.conf) and writes two output files (Summary-Results and Summary-Processes).
- To run the program, enter the following in the command line.

```
$ java Scheduling scheduling.conf
```

- The program will display "Working..." while the simulation is working, and "Completed." when the simulation is complete.
- The simulator reads parameters from the configuration file ("scheduling.conf").

2.1 The configuration file

- The configuration file (scheduling.conf) is used to specify various parameters for the simulation.
- There are a number of options which can be specified in the configuration file. These are summarized in the table below.

Keyword	Values	Description
numprocess	n	The number of processes to create for the simulation.
meandev	n	The average length of time in milliseconds that a process should execute before terminating.
standdev	n	The number of standard deviations from the average length of time a process should execute before terminating.
process	n	The amount of time in milliseconds that the process should execute before blocking for input or output. There should be a separate process directive for each process specified by the numprocess directive.
runtime	n	The maximum amount of time the simulation should run in milliseconds.
timeslice	n	The amount of time for a timeslice in the round-robin implementation.
test	1 or 0	If testing == 1, system use meandev as each process's length. This is for testing purposes.

3 FIFO Scheduling

- Open the 'SchedulingAlgorithm.java' file. You can find the implementation of the FIFO scheduling algorithm here.
- Please go through the FIFO method carefully and try to understand how this version of FIFO algorithm works. Specifically, give attention to the timer-counter usage in the implementation.

- Try to identify the design decisions taken when implementing this algorithm. I.e: assumptions, scope and limitations.
- Change values in the configuration file and try different types of workloads. Set 'test' to '0' when you are experimenting.

4 Round-robin Scheduling

- Implement the round-robin scheduling algorithm in the 'SchedulingAlgorithm.java' file. Use the given '.test' output files as references.
- Assume that the time each process stays in the blocked state is 0.
- Use the print functions in the 'SchedulingAlgorithm.java' file to update the 'Summary-Processes' file.

5 Testing

- After implementation, execute the 'run_test.py' to check the accuracy of your algorithm. Please note that this script is given for your convenience. It is not mandatory to use this. It is written in python and requires a working installation of python3 to run.
- It compares your output file with a preprocessed output file, which is generated with a round-robin scheduling algorithm.
- You can add more of your own tests if you prefer. Please add those new tests to 'tests.txt' file.

6 Submission

- Submit the 'SchedulingAlgorithm.java' file to the submission link in the moodle before the deadline.
- You will be marked against a set of test cases.
- Please keep the code clean and add comments. There will be marks for the code quality and comments.

7 Assessment

Your submission will be tested against input that we have designed. To help you get started, a file called run_test.py has been supplied. Keep the following points in mind:

- **run_test.py is only a sample.** The actual test will contain more tests.
- **Do NOT** change the input/output format of the code given to you. Any change will result in your code failing the tests.
- **Do NOT** output anything other than what has been asked for. If you have added any outputs for your own convenience, you should remove/comment them **before** submission.