CSC 428 – Operating Systems
Programming Assignment #1 – CPU Scheduling Simulator
Due on Wednesday, September 27, by 8:00 PM

Overview:
For this project, you will be writing a program to implement and model a CPU running various scheduling policies that we have discussed in class. Your program will read the information about the processes from a text file, then run the simulation and report the data about that simulation. You can complete this project in any programming language you choose; if you want to use a language other than Java, C++, or Python, please check with me first. You should submit your code files (only the source files) on Moodle; alternatively, a link to a Replit is also acceptable.

Assignment:
You should write a program that does the following:

- Welcomes the user to your program.
- Prompts the user to enter a file name containing process data (see file format below).
- If the file does not exist, or consists of an invalid format, print an error message and exit the program.
- Otherwise, open the file and read in the data.
- Prompt the user which algorithm they want to simulate (see below).
- Simulate the appropriate algorithm and report the data about the simulation (see below).

Once the simulation is complete and data has been reported, the program can end.

File Format:
The file containing the information about the processes will be in a specific format, descibed here:

- There will be some number (greater than or equal to 1) of lines; each line will contain information about a single process. The line will consist of:
    - The name of the process (a string of at least one character)
    - The arrival time of the process (an integer)
    - The length of the process (an integer)
- After all the process information, there will be a single line consisting of the string "END".

You can assume that no process will have the name "END". You can also assume that the processes will be listed in the file in order by arrival time. Finally, you can assume that every process has a unique arrival time (i.e., no two processes will arrive at the same time).

Algorithms:
Your program must offer the user a choice of algorithms to simulate. The algorithms you can choose from are FIFO (first-in, first-out), SJF (shortest job first), STCF (shortest time to completion

first), or RR (round robin). If you simulate RR, your time quantum should be 4.

Your program should simulate two or three of these algorithms, satisfying these restrictions:

- If you choose to simulate two algorithms, one of the algorithms you simulate must be RR, and you may not select FIFO as one of your algorithms. So, basically you must implement SJF or STCF along with RR.
- If you choose to simulate three algorithms, you can select any of the algorithms you wish.

Data Report:

Once the simulation is complete, your program should report the data about the simulation. This reporting should consist of the following:

- The CPU usage chart. This should consist of list of symbols representing which processes were run for each second of operation. For example, if process A starts at 0 and runs for 5 second, then B runs for 4 seconds, then C for 2 seconds, the CPU usage chart that would be printed should be:

  ```
  A A A A A B B B B C C
  ```
- After the CPU usage chart, you should print the following information about each process (this information should be on a single line, and clearly labeled):
  - The name of the process
  - The response time for that process
  - The turnaround time for that process
- At the end, also print the average response time and average turnaround time.

Specifications:

The following specifications are required for this project. Each specification is graded on the 0-1 scale, and your project total score will be equal to the total number of points from the specifications.

1. Your program represents a sincere attempt to solve the problem.
2. Your program compiles and runs, assuming specification 1 is met.
3. Your program greets the user and prompts the user to enter a file name.
4. Your program successfully ends if the file does not exist or cannot be read properly.
5. Your program successfully reads the data from the file name given.
6. Your program successfully prompts the user to select an algorithm.
7. Your program implements FIFO correctly, if FIFO was selected for implementation.
8. Your program implements SJF correctly, if SJF was selected for implementation.
9. Your program implements STCF correctly, if STCF was selected for implementation.
10. Your program implements RR correctly, if RR was selected for implementation.
11. Your program correctly prints the CPU usage chart at the end.
12. Your program correctly prints the process data for each process at the end.
13. Your program correctly calculates and prints the average response and turnaround times.
14. Your program is well-written and styled (formatting, consistent indentation, comments, etc.).