Option 1 (**20 + *2 bonus points***)

You (and your team if you decided to work in a team) should use the Java IDE you find appropriate (e.g., NetBeans, Eclipse) to solve this project. You are building a round robin preemptive scheduler to organize the work of processes in a computing system. Any process can be either newly created, ready to be selected by the scheduler, running for a certain amount of time (a.k.a. Quantum), or terminated when done. You may assume that a process will not be waiting/blocked while being in the running state. Your scheduler is composed of the following components:

Process (Component 1): Create a class that represents a process in a computing system. Any process, in order to work within your scheduler, needs to be described in terms of: 1) *Unique identifier* (integer value); 2) *Arrival time* (integer value represents when the process became available in the system); 3) *Execution time* (integer value represents how long the process should run to finish the task); and 4) *Priority* (integer value represents the priority of the process with 1 represents the highest priority). The class must have the following mandatory functions:

1- Default constructor: provides a unique ID and initializes the rest of variable by zeros.
2- Full constructor: initializes the four variables by the constructor's data.
3- Get/Set for each variable.
4- toString: return a string with the current values of the process' variables.

You can add more functions if required.

Scheduler (Component 2): Function that accepts an array of processes and quantum (integer value represents the maximum time window for how long each process should run). The scheduler schedules the input processes with respect to their arrival time and considers their unique identifiers whenever they have same arrival time (two processes with same arrival time, then consider the one with smaller id as first choice). The scheduler finally prints the selected processes for running by calling the toString function.

Initializer (Component 3): As mentioned in the previous component, the scheduler accepts an array of processes as input. Initializer, function, creates such an array of processes dynamically. The initializer reads an input text file that resides on your computer and holds information about the processes you need to create for your system. The file is structured as follows:

```
Number of processes, quantum time
id, arrival time, executing time, priority
id, arrival time, executing time, priority
…..
id, arrival time, executing time, priority
```

The first line of the file shows the number of processes you have in the file and the quantum time (the input to the scheduler). The next lines represent information about the different processes. Take the following as an example for such file – here we have 3 processes and quantum to be 5. Each process (takes a separate line) is represented as id, arrival time, and executing time. For example: first process has id of 1, arrival time of 5, execution time of 8, and priority of 1 (highest priority).

```
3, 5
1, 5, 8, 1
2, 0, 10, 2
3, 10, 7, 3
```

Main Engine (Component 4): The engine is the module that runs the different components you developed sequentially. As your scheduler is just one module of the OS, it should run in parallel with other modules of the OS. The engine works on two threads, thread 1 and thread 2, that run sequentially. Thread 1 runs ***the initializer***, and Thread 2 waits until thread 1 is done then runs the ***scheduler directly after***.

```
public class MainEngine {

        //array of processes, initially empty

        //quantum value, initially zero

        public static void Initializer (…) {

        }
        public static void Scheduler (…) {

        }
        public static void main (String[] args) {

           Thread 1: run the Initializer

           Thread 2: run the scheduler

        }
}
```

Project submission Guidelines:

1. Submit only ONE ZIP to the folder titled FinalProject under the D2L Assignments tab (other formats will not be accepted). ***Only one of the team members should submit the developed team project***.

2. The .ZIP file contains:
   - The java files you developed on.
   - PDF file constructed as follows:
     - Page1: your name and the name of your teammate.
     - Page2: the output (screenshots) of running your scheduler using the shared testcases (0,1, and 2)
     - Page3: the two (at least) test cases you developed to test the correctness of your scheduler
     - Page4: the output (screenshots) of running your scheduler using your testcases.

3. The submission is due **11:59pm – May 5th**. You can submit any updates to your submission within 12 hours after this due date will be graded out of 75% of the final project's grade (for the whole team). After this grace period your late submission will not be accepted.