Assignment 1
CPU Scheduler Simulation

Assignment Description:

In this assignment, you will develop a CPU scheduler simulation program that simulates the behavior of different scheduling algorithms. This assignment will help you deepen your understanding of scheduling algorithms and their impact on system performance in operating systems.

Requirements:

1. **Scheduling Algorithms**: Implement the following five CPU scheduling algorithms:
   - First-Come, First-Served (FCFS) (for batch systems)
   - Shortest Job First (SJF) (for batch systems)
   - Priority Scheduling (for batch systems)
   - Round Robin (RR) (for interactive systems)
   - Priority scheduling + RR (for interactive systems)

2. **Process Representation**: Define a data structure to represent processes. Each process should have attributes such as process ID, arrival time, burst time (required CPU time), priority (if applicable), etc. You can use a data structure similar to the process table seen in class.

3. **Input Data**: Generate or read input data consisting of a set of processes with their respective attributes (arrival time, burst time, priority, etc.). Your code should support generating random processes, and reading them from a file.

4. **Scheduler Implementation**: Implement each scheduling algorithm using appropriate data structures and algorithms. Your scheduler should schedule processes based on the chosen algorithm and update the CPU's state accordingly.

5. **Visualization**: Provide a graphical or textual representation of the scheduling process. You can display information such as the order in which processes are scheduled, the remaining burst time for each process, the turnaround time, waiting time, etc. Visualization will help in understanding how each scheduling algorithm works and how it affects system performance.

6. **Performance Metrics**: Calculate and display performance metrics for each scheduling algorithm, including average turnaround time, average waiting time, CPU utilization, etc. Compare the performance of different algorithms under various scenarios.

7. **User Interaction**: Allow users to input parameters such as the number of processes, burst time range, arrival time range, etc., and choose which scheduling algorithm to simulate. Provide options for users to customize the simulation according to their preferences.

8. **Documentation**: Include comments throughout your code to explain the purpose of each function and significant sections of code. Additionally, provide a brief README file that explains how to compile and run your program, as well as any additional features or design decisions you made. The readme file should be part of your submitted report.

9. **Testing**: Test your program with different sets of input data and scheduling algorithms to ensure its correctness and effectiveness in simulating CPU scheduling. Include sample testing into your report.

Submission Guidelines:

1. Submit your source code files along with any necessary build scripts or configuration files.

2. Include a README file that provides instructions for compiling and running your program, as well as any additional information you think is necessary for understanding your implementation.

3. Make sure your code is well-documented and organized.

4. Submit a report detailing your design choices, implementation decisions, usage scenarios, testing cases, and performance comparisons of the implemented algorithms.

Marking Scheme:

1. Scheduling Algorithm Implementation (25 points):
   - Each implemented scheduling algorithm (FCFS, SJF, Priority, RR, etc.)
   - Correctness and efficiency of the algorithm implementations.

2. Process Representation (10 points):
   - Design and implementation of the data structure to represent processes.
   - Appropriateness of attributes included (process ID, arrival time, burst time, priority, etc.).

3. Input Data Handling (10 points):
   - Ability to generate and read input data consisting of processes.

    - Handling of input data errors and edge cases.

4. Scheduler Implementation (20 points):
    - Correctness and effectiveness of the scheduling algorithm implementations.
    - Proper updating of CPU's state based on scheduling decisions.

5. Visualization (10 points):
    - Clarity and effectiveness of graphical or textual representation of the scheduling process.
    - Informativeness of the visualization in understanding the behavior of each algorithm.

6. Performance Metrics Calculation (10 points):
    - Accuracy of performance metrics calculation (average turnaround time, average waiting time, CPU utilization, etc.).
    - Ability to compare and analyze the performance of different scheduling algorithms.

7. User Interaction (5 points):
    - User-friendliness of the interface.
    - Flexibility in allowing users to customize the simulation parameters.

8. Documentation (5 points):
    - Clarity and completeness of code comments.
    - Adequacy of the README file in explaining how to compile and run the program and any additional features or design decisions.

9. Testing (5 points):
    - Thoroughness of testing with different sets of input data and scheduling algorithms.
    - Ensuring correctness and effectiveness of the simulation under various scenarios.

Total: 100 points

Bonus points (up to 5 points):
- Additional features beyond the basic requirements.
- Innovative approaches in algorithm design or visualization.
- Exceptional clarity and organization of code.