Kathryn Sarullo

**Round Robin (RR):**

The main function includes the following. It asks for the number of processes, the burst time of each process is, and the quantum time. It then calls the function, roundrobin, passing in each of these values.

The function, roundrobin, then initializes all the variables we will need later such as the waiting and turn around time for each process, total wait and turnaround times to calculate the averages of each, t for time, and the amount of burst remaining for each process. First, we make a copy of the burst times, so we can keep track of the amount of time remaining for each process. Then the scheduling process starts. For every process, if there is still burst time left, run for the quantum time. Repeat this process until every process has no time left on its burst time remaining. At the end of this function, a table is printed out including each process, its burst time, the waiting time, and turn-around time. Underneath is the average waiting time and average turn-around time for these specific processes and burst times.

**Earliest Deadline First (EDF):**

The main function includes the following. It asks for the number of processes, the burst time of each process is, and the deadline time. Next, it sorts the burst times and periods by the shortest period. It then calls the function, EDF, passing in each of these values.

The function, EDF, checks what deadline is next and runs that process until it finishes. If it finishes before the deadline, it picks up the next process. At the time of the deadline, it checks to see which deadline is next and prioritizes that process. It does this a specified number of times. Once it reaches that count, it prints out a table including each process, its burst time, its deadline time, the waiting time, and turn-around time. Underneath is the average waiting time and average turn-around time for these specific processes and burst times.

Kim Chen

**First Come First Serve (FCFS):**

This simulation was implemented in Python. It takes in a command line argument that is used to determine the number of processes to create. Process creation involves appending the name and the burst time of the process to a list. Burst times are determined by randomly selecting a number from 1 – 25. This list is then shuffled to simulate different arrival times for each process. The program then iterates through the array and calculates the waiting time and turnaround time for each process to simulate execution. Finally, the average waiting time and average turnaround tie are calculated.

**Lottery Scheduling:**

This simulation was implemented in Python. It takes in a command line argument that is used to determine the number of processes to create. Process creation involves appending the name and the burst time of the process to a list. Burst times are determined by randomly selecting a number from 1 – 25. 2 other lists are also created to hold the waiting times and burst times of each process for later use in calculating the average waiting times and average turnaround times. The quantum was arbitrarily selected to be 8.

The program goes through and checks the remaining burst times of each process. If a process’ burst time is less than or equal to zero, it has finished executing and therefore can be removed from the process queue. If the process queue is empty we have finished executing all processes. If there is only one process left, then the scheduler executes the process without assigning tickets. Otherwise, we handout tickets to each process. This implementation prioritizes processes with smaller burst times, and the maximum amount of tickets a process can receive is equal to the maximum value of our randomly selected burst times, in this case it would be 25. After all tickets have been handed out, a random ticket is selected, and that process get executed for the quantum time or until it’s burst time is 0. Once that process has finished running, new tickets are distributed and a new winner is selected. This process continues until all processes have finished executing. The average waiting time and average turnaround time are then calculated.

Performance Analysis

Average Waiting Time

Average Turnaround Time

For this analysis, each simulation was run 100 times for 10, 50, 100, and 200 processes, and the waiting time and turnaround time were calculated from the average of all these runs. We standardized the burst times to be random numbers between 1 – 100. For the simulations that needed it, the quantum number was kept at 40.

As seen from the graphs shown below, overall Round Robin Scheduling performed the worst while Lottery scheduling performed the best. Round Robin’s performance can be explained through its implementation, specifically in regard to the quantum time. The quantum number was selected to be 40, which causes the scheduler to perform more context switching. If our comparisons had select the quantum number to be closer to 80, as stated by the rule of thumb, we would likely see waiting and turnaround times to be closer to the First Come First Serve Scheduler.