**ASSIGNMENT 1: SCHEDULING ALGORITHMS**

**Name:** Sravya Ganne (201101201), Vaibhavi Desai (201101209)

We have implemented three scheduling algorithms. Namely

* First Come first serve(fcfs.m)
* Shortest job first(sjf.m)
* Round robin(rr.m)

**Running the program:**

Follow the steps to run the program.

* Write the following commands to run the respective programs in matlab console.

fcfs

sjf

rr

Below is a detailed explanation of what each code does.

**First come first serve:**

1. The first come first scheduling algorithm runs the process that comes first, finishes it and then starts the next process in the queue.
2. In the code the burst times of processes are hard coded in the array 'btime'.
3. Then the wait time of each process can be found by adding the waiting time and burst time of process preceding it.
4. These values are stored in an array 'wtime'.The total waiting time 't1' can be found by adding the waiting times of individual processes.
5. The turnaround time is found using adding the burst time and waiting time of each process and it is stored in the array 'tatime'.
6. 't2' is the total turnaround time found by adding individual turnaround times.

**Shortest job first:**

1. The shortest job first algorithm runs the process whose burst time is lowest.
2. In the code we are assuming all the processes have arrived already.
3. The burst times are hard coded in the array 'ptime'.
4. As all the processes have arrived and burst times are known they are sorted using 2 'for' loops.
5. Then the wait time of each process can be found by adding the waiting time and burst time of process preceding it.
6. These values are stored in an array 'wtime'.The total waiting time 'total' can be found by adding the waiting times of individual processes.
7. The turnaround time is found using adding the burst time and waiting time of each process and it is stored in the array 'tatime'.
8. 'total2' is the total turnaround time found by adding individual turnaround times.

**Round Robin:**

1. The round robin algorithm runs a process for a given quantum time. If the process gets completed in the given time it is dequeued else it is put at the end of the queue so that it can run again later.
2. In the code the burst time and quantum time are hard coded in array 'btime' and variable 'q' respectively.
3. Initially the remaining time array 'rtime' is equated to the burst time as no process has run yet.
4. A process which is present in the starting of the array is selected and its remaining time is compared with the quantum time.
5. If remaining time (rtime) of process is greater than the selected process is run for quantum time(q) and this q is subtracted from rtime. For other processes q is added to waiting time which is an array 'wtime'.
6. If quantum time is greater the process is run to completion and rtime is set to 0 for the process selected. For other processes q is added to waiting time.
7. There is also a 'flag' variable which is set if the rtime of one or more processes is greater than zero.
8. So if the flag is set the 4, 5, 6 steps are repeated.
9. The turnaround time is found using adding the burst time and waiting time of each process and it is stored in the array 'tatime'.
10. The variables 'b' and 't' store values of total waiting time and total turnaround time which is found by summing values of all the processes.