CS-266 Mid Sem Lab Exam

Roll no:- 201951105 Name:- Nishant Andpriya

In this assignment, you have to implement a hybrid scheduling algorithm which would be the combination of Round-Robin and Shortest Job First.

You have to take input from the user as

1. No of processes
2. provide process id
3. Burst time (minimum value should be at least 20)
4. Your Name
5. Your roll No

Code:-

```
import java.util.*;

public class preendsem {

    static void FATRR(int proc[], int n, int bt[], int wt[], int quantum) {
        int temp_wt[] = new int[n];
        for (int i = 0; i < n; i++) {
            bt[i] -= quantum;
            if (i != 0) {
                temp_wt[i] = temp_wt[i - 1] + quantum;
                wt[i] = wt[i - 1] + quantum;
            }
        }
        FWTSJR(proc, n, bt, wt);

        System.out.println("Processes " + " Burst time " + " Waiting time " + " T
urn around time");
        int total_wt = 0, total_tat = 0;
        int tat[] = new int[n];
        for (int i = 0; i < n; i++) {
            total_wt = total_wt + wt[i];
            total_tat = total_tat + tat[i];
            System.out.println("      " + (i + 1) + "        " + bt[i] + "      "
 + temp_wt[i] + "       " + tat[i]);
        }
    }

    static void FWTSJR(int proc[], int n, int bt[], int wt[]) {
        int rt[] = new int[n];
```

```
        for (int i = 0; i < n; i++)
            rt[i] = bt[i];

        int complete = 0, t = 0, minm = Integer.MAX_VALUE;
        int shortest = 0, finish_time;
        boolean check = false;

        while (complete != n) {

            for (int j = 0; j < n; j++) {
                if ((rt[j] < minm) && rt[j] > 0) {
                    minm = rt[j];
                    shortest = j;
                    check = true;
                }
            }

            if (check == false) {
                t++;
                continue;
            }

            rt[shortest]--;

            minm = rt[shortest];
            if (minm == 0)
                minm = Integer.MAX_VALUE;

            if (rt[shortest] == 0) {

                complete++;
                check = false;

                finish_time = t + 1;

                wt[shortest] += finish_time - bt[shortest];

                if (wt[shortest] < 0)
                    wt[shortest] = 0;
            }

            t++;
        }
    }
```

```
    static void FTAT(int processes[], int n, int bt[], int wt[], int tat[]) {

        for (int i = 0; i < n; i++) {
            tat[i] = bt[i] + wt[i];
        }
    }

    static void FAT(int processes[], int n, int bt[], int quantum) {
        int wt[] = new int[n], tat[] = new int[n];
        int total_wt = 0, total_tat = 0;

        FATRR(processes, n, bt, wt, quantum);

        FTAT(processes, n, bt, wt, tat);

        System.out.println("Processes " + " Burst time " + " Waiting time " + " T
urn around time");

        for (int i = 0; i < n; i++) {
            total_wt = total_wt + wt[i];
            total_tat = total_tat + tat[i];
            System.out.println(" " + (i + 1) + " " + bt[i] + " " + wt[i] + " " +
tat[i]);
        }

        System.out.println("Average waiting time = " + (float) total_wt / (float)
 n);
        System.out.println("Average turn around time = " + (float) total_tat / (f
loat) n);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n, q;
        System.out.println("Enter no of process");
        n = sc.nextInt();
        int processes[] = new int[n];
        int burst_time[] = new int[n];

        for (int i = 0; i < n; i++) {
            System.out.print("Enter process id for process" + (i + 1) + " : ");
            processes[i] = sc.nextInt();
            System.out.print("Enter burst time for process" + (i + 1) + " : ");
            burst_time[i] = sc.nextInt();
```

```
        }
        System.out.println("Enter name without roll spaces");
        String s = sc.next();
        System.out.print("Enter your roll number : ");
        int roll = sc.nextInt();
        q = GQ(s, roll);
        System.out.println("Quantum is =  " + q);

        FAT(processes, n, burst_time, q);
    }

    public static int GQ(String s, int roll) {
        int ans = GSascii(s);
        ans %= 20;
        if (ans != 0) {
            return ans;
        } else {
            ans += GSrollno(roll);
            ans %= 20;
            if (ans != 0) {
                return ans;
            } else {
                return (int) (Math.random() * 19 + 1);
            }
        }
    }

    public static int GSrollno(int m) {
        int n, sum = 0;
        while (m > 0) {
            n = m % 10;
            sum = sum + n;
            m = m / 10;
        }
        return m;
    }

    public static int GSascii(String s) {
        int sum = 0;
        for (int i = 0; i < s.length(); i++)
            sum += s.charAt(i);
        return sum;
    }
}
```

Output:-

```
D:\vs code files\java files\sem 4 assigment\CS266assigment> d: && cd "d:\vs code files\java files\sem 4 assigment\CS266assigment" &
-XX:+ShowCodeDetailsInExceptionMessages -Dfile.encoding=UTF-8 -cp C:\Users\Lenovo\AppData\Roaming\Code\User\workspaceStorage\b5ebb
Enter no of process
3
Enter process id for process1 : 1
Enter burst time for process1 : 20
Enter process id for process2 : 2
Enter burst time for process2 : 15
Enter process id for process3 : 3
Enter burst time for process3 : 25
Enter name without roll spaces
nishantandoriya
Enter your roll number : 201951105
Quantum is =  12
Processes  Burst time  Waiting time  Turn around time
      1        8      0       0
      2        3      12       0
      3        13      24       0
Processes  Burst time  Waiting time  Turn around time
 1 8 3 11
 2 3 12 15
 3 13 35 48
Average waiting time = 16.666666
Average turn around time = 24.666666
```