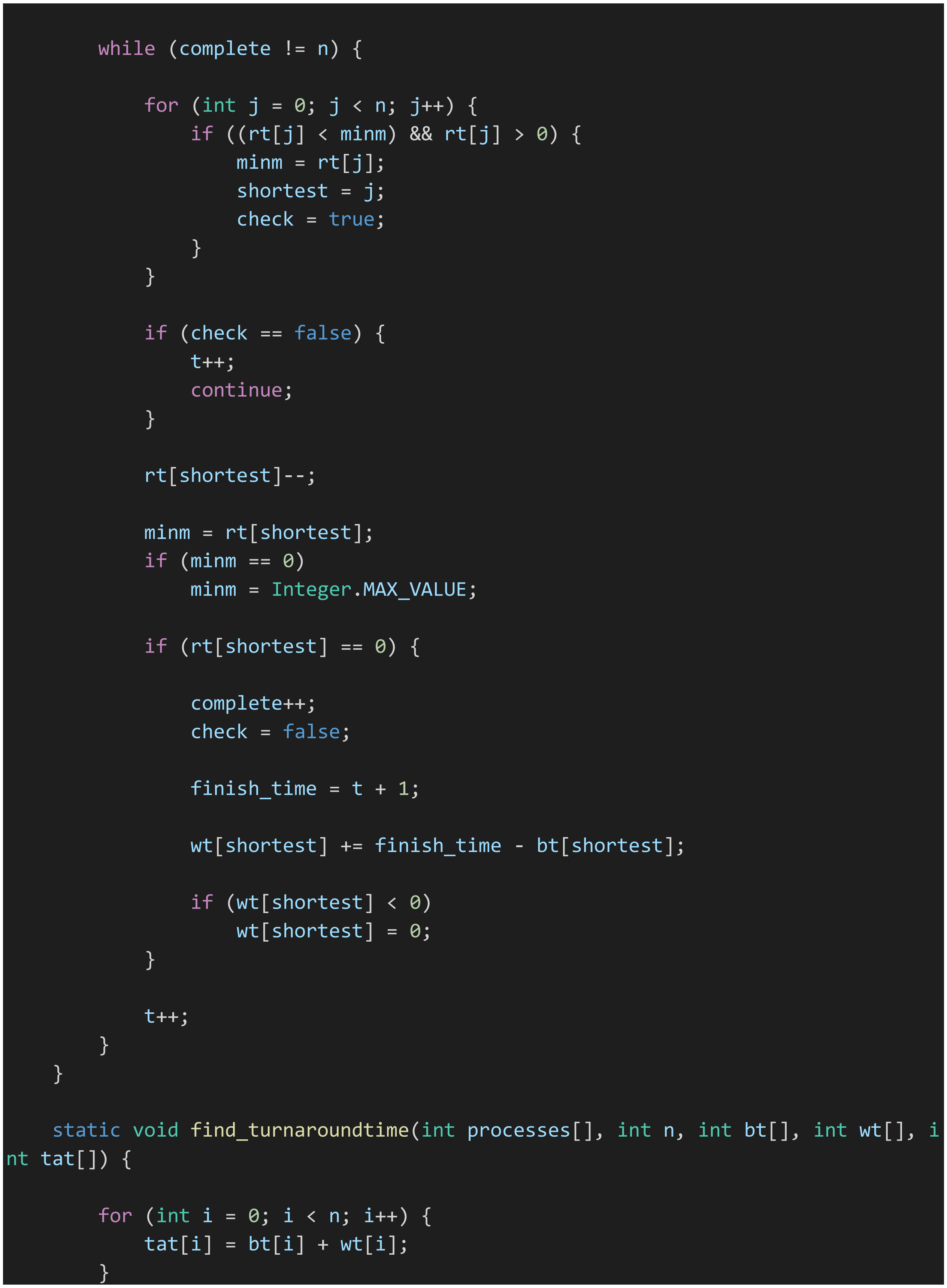
CS-266 Mid Sem Lab Exam

Roll no:- 201951134 Name:- Sameer Anand

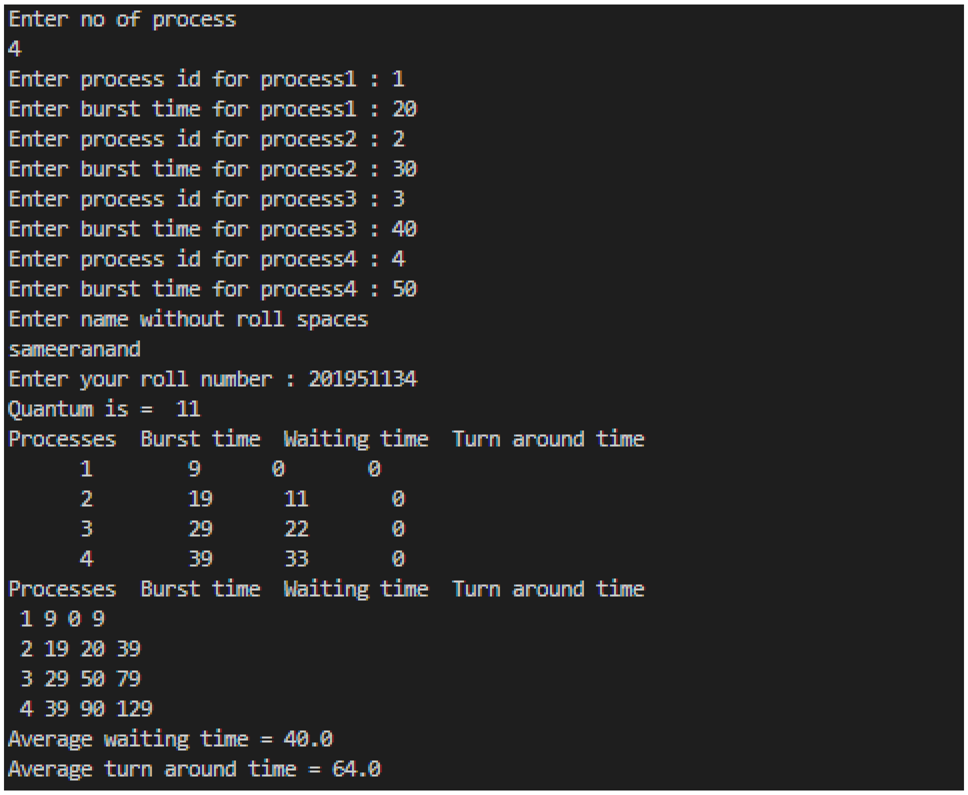
Code:-

Output:
```
import java.util.*;


public class preendsem {

    static void find_avgtime_rr(int proc[], int n, int bt[], int wt[], int quantu
m) {
        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;
            }
        }
        findavgwatingtimeSJr(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 findavgwatingtimeSJr(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 find_turnaroundtime(int processes[], int n, int bt[], int wt[], i
nt 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;

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

        find_turnaroundtime(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);
```

```java
        System.out.println("Quantum is =  " + q);

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

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

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

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



```
Enter no of process
4
Enter process id for process1 : 1
Enter burst time for process1 : 20
Enter process id for process2 : 2
Enter burst time for process2 : 30
Enter process id for process3 : 3
Enter burst time for process3 : 40
Enter process id for process4 : 4
Enter burst time for process4 : 50
Enter name without roll spaces
sameeranand
Enter your roll number : 201951134
Quantum is =  11
Processes  Burst time  Waiting time  Turn around time
      1        9      0       0
      2        19      11       0
      3        29      22       0
      4        39      33       0
Processes  Burst time  Waiting time  Turn around time
 1 9 0 9
 2 19 20 39
 3 29 50 79
 4 39 90 129
Average waiting time = 40.0
Average turn around time = 64.0
```