|  |
| --- |
| [Type the company name] |
| CS262 |
| Lab 6 |
|  |
|  |
|  |

|  |
| --- |
| SAMEER ANAND  201951134 |

**Indian Institute of Information Technology Vadodara**

**Problem 1**

In this assignment, you have to implement the round-robin scheduling strategy to schedule the processes. You have to create two programs.

**Program 1:**

• This will create a file that will be input for the second program.

• Save this program as program1.

• You have to write the code such that it will ask the user to enter

**–** Number of processes (maximum 5)

**–** Process id (as String)

**–** Process arrival of each process (0-5)

**–** Processing time for each process (15-30) (integer)

**–** The elapsed time between I/O interrupts (system calls) (1-3) (float),

**–** Time spent in waiting and processing the I/O (1-5) (float)

**–** Priority for each task (same or different) (1-5)

**–** The created input file should be saved in test.dat.

The input file should look like this:

P1 0 20 1.5 2.0 1

P2 2 15 1.0 2.5 2

P3 4 18 2.3 3.0 1

P4 2 27 2.5 2.5 2

P5 5 24 1.0 4.0 3

**Code:-**

import java.io.FileOutputStream;

import java.io.FileWriter;

import java.io.OutputStream;

import java.util.\*;

public class lab6\_q1 {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int no\_process = 0;

        try {

            OutputStream out = new FileOutputStream(

                    "D:\\vs code files\\java files\\sem 4 assigment\\CS266assigment\\lab6\_q1.dat");

        } catch (Exception e) {

            System.out.println("Exception caught " + e);

        }

        System.out.println("Number of processes (maximum 5)");

        no\_process = sc.nextInt();

        String[] pid = new String[no\_process];

        for (int i = 0; i < no\_process; i++) {

            int temp = i;

            temp += 1;

            System.out.println("Process id of process " + temp);

            pid[i] = sc.next();

        }

        int[] arrival = new int[no\_process];

        for (int i = 0; i < no\_process; i++) {

            int temp1 = i;

            temp1 += 1;

            System.out.println("Enter Process arrival of process" + temp1);

            arrival[i] = sc.nextInt();

        }

        int[] processing = new int[no\_process];

        for (int i = 0; i < no\_process; i++) {

            int temp2 = i;

            temp2 += 1;

            System.out.println("Enter Processing time for process " + temp2);

            processing[i] = sc.nextInt();

        }

        float[] io = new float[no\_process];

        for (int i = 0; i < no\_process; i++) {

            int temp3 = i;

            temp3 += 1;

            System.out.println("Enter The elapsed time between I/O interrupts of process " + temp3);

            io[i] = sc.nextFloat();

        }

        float[] iowait = new float[no\_process];

        for (int i = 0; i < no\_process; i++) {

            int temp4 = i;

            temp4 += 1;

            System.out.println("Enter Time spent in waiting and processing the I/O for process " + temp4);

            iowait[i] = sc.nextFloat();

        }

        int[] priority = new int[no\_process];

        for (int i = 0; i < no\_process; i++) {

            int temp5 = i;

            temp5 += 1;

            System.out.println("Enter Priority for process " + temp5);

            priority[i] = sc.nextInt();

        }

        String[][] write\_arr = new String[no\_process][6];

        for (int i = 0; i < no\_process; i++) {

            write\_arr[i][0] = pid[i];

        }

        for (int i = 0; i < no\_process; i++) {

            write\_arr[i][1] = Integer.toString(arrival[i]);

        }

        for (int i = 0; i < no\_process; i++) {

            write\_arr[i][2] = Integer.toString(processing[i]);

        }

        for (int i = 0; i < no\_process; i++) {

            write\_arr[i][3] = Float.toString(io[i]);

        }

        for (int i = 0; i < no\_process; i++) {

            write\_arr[i][4] = Float.toString(iowait[i]);

        }

        for (int i = 0; i < no\_process; i++) {

            write\_arr[i][5] = Integer.toString(priority[i]);

        }

        try {

            FileWriter fw = new FileWriter(

                    "D:\\vs code files\\java files\\sem 4 assigment\\CS266assigment\\lab6\_q1.dat");

            for (int i = 0; i < no\_process; i++) {

                for (int j = 0; j < 6; j++) {

                    fw.write(" " + write\_arr[i][j] + " ");

                }

                fw.write("\n");

            }

            fw.close();

        } catch (Exception e) {

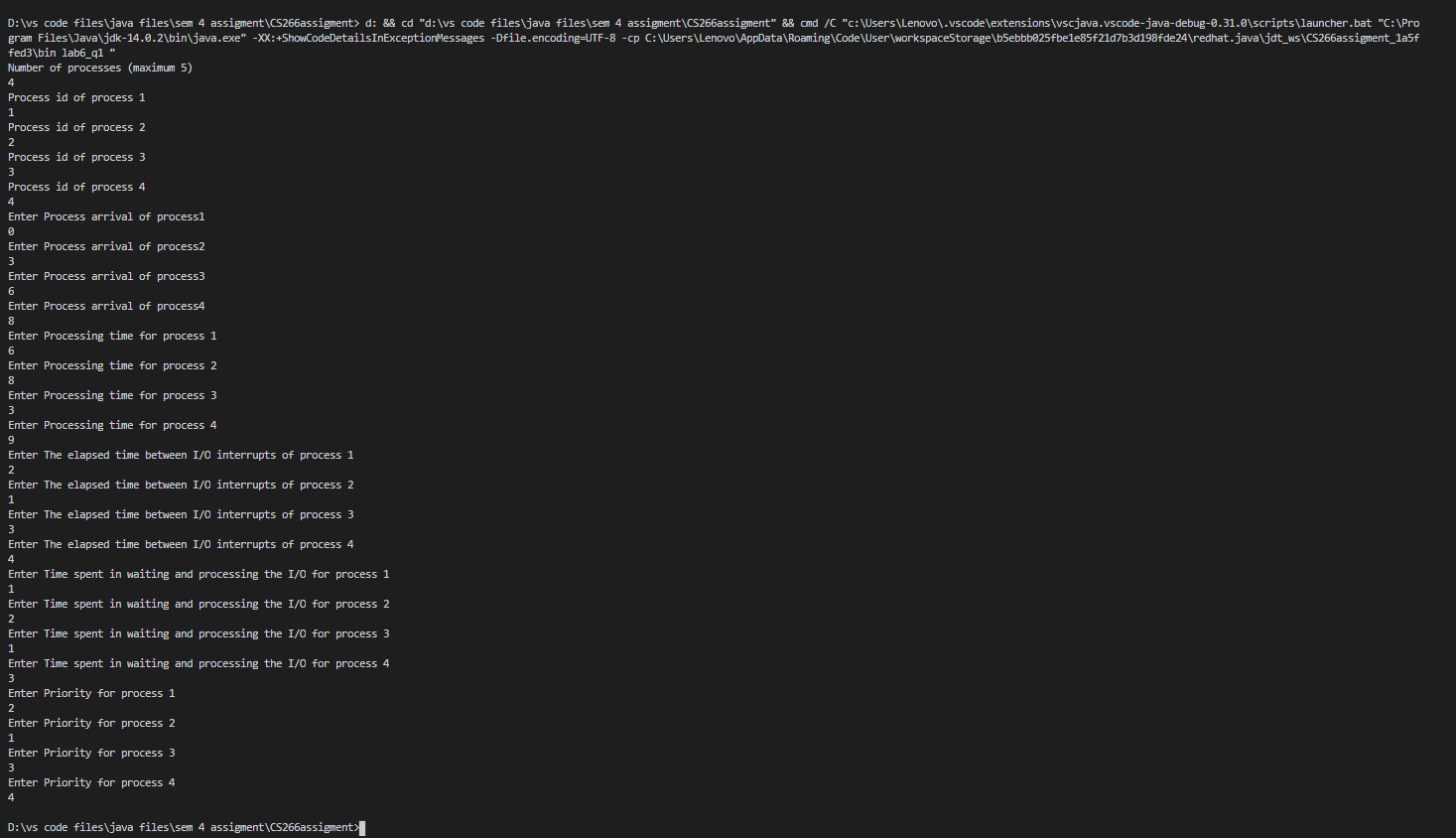
            System.out.println("Exception caught: " + e);

        }

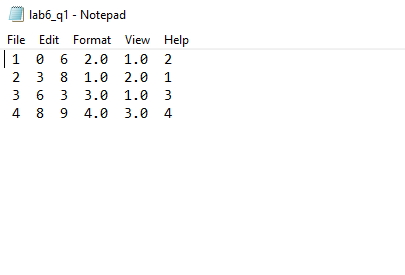
    }

}

**Input for this program:-**



**Output:-**



**Program 2:**

• Save this program as program2

• Second program will read the file (test.dat file) that consists of a list of processes with the desired parameters for each process.

• Bonus marks (+1) if you pass the test.dat file as a command-line argument, otherwise you can read the test.dat file in your second program ( But NO bonus marks for this).

• This program will implement the RR algorithm.

• The program will simulate the execution of the processes and provide the following: **–** turnaround time of each process,

1

**–** waiting time of each process,

**–** average turnaround time,

**–** average waiting time.

• The time slice can be varied from 1 to 10sec. Then plot the graph that will show how

**–** the average turnaround time for processes varies with time slice/quantum. **–** how the average waiting time for processes varies with time slice/quantum.

**Code:**-

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.util.\*;

public class lab6\_q2 {

    public static void main(String[] args) {

        System.out.println("Entered program2");

        File ogFile = null;

        for (String str : args) {

            File newFile = new File(str);

            if (newFile.exists()) {

                ogFile = newFile;

            }

        }

        String str = null;

        String temp\_str = "";

        try {

            BufferedReader br = new BufferedReader(new FileReader(ogFile));

            while ((str = br.readLine()) != null) {

                temp\_str += str;

            }

        } catch (Exception e) {

            System.out.println("Excpetion caught: " + e);

        }

        int cnt = 0;

        int len = temp\_str.length();

        String[] stringof = new String[len];

        for (int i = 0; i < len; i++) {

            stringof[i] = "";

        }

        int j = 0;

        temp\_str = temp\_str.trim();

        int k = 0;

        outer: while (k < 30) {

            if (j == temp\_str.length()) {

                return;

            }

            try {

                while (temp\_str.charAt(j) != ' ') {

                    stringof[k] += temp\_str.charAt(j);

                    j += 1;

                }

            } catch (Exception e) {

                System.out.println("Exception ignored");

            }

            j += 2;

            k += 1;

        }

        for (String x : stringof) {

            System.out.println(x);

        }

        int j\_arr = 1;

        int j\_burst = 2;

        int j\_ioi = 3;

        int j\_iow = 4;

        int total = 0;

        int count\_i = 0;

        while (stringof[count\_i] != "") {

            total += 1;

            count\_i++;

        }

        int total\_temp = total / 6;

        float[] arrival = new float[total\_temp];

        float[] burst = new float[total\_temp];

        float[] ioint = new float[total\_temp];

        float[] iowait = new float[total\_temp];

        float[] exit = new float[total\_temp];

        float[] wait = new float[total\_temp];

        float[] turnaround = new float[total\_temp];

        /\* int[] wait\_time=new int[total\_temp]; \*/

        for (int i = 0; i < total\_temp; i++) {

            arrival[i] = Float.parseFloat(stringof[j\_arr]);

            j\_arr += 6;

            if (stringof[j\_arr] == "") {

                break;

            }

        }

        for (int i = 0; i < total\_temp; i++) {

            burst[i] = Float.parseFloat(stringof[j\_burst]);

            j\_burst += 6;

            if (stringof[j\_burst] == "") {

                break;

            }

        }

        for (int i = 0; i < total\_temp; i++) {

            ioint[i] = Float.parseFloat(stringof[j\_ioi]);

            j\_ioi += 6;

            if (stringof[j\_ioi] == "") {

                break;

            }

        }

        for (int i = 0; i < total\_temp; i++) {

            iowait[i] = Float.parseFloat(stringof[j\_iow]);

            j\_iow += 6;

            if (stringof[j\_iow] == "") {

                break;

            }

        }

        /\*

         \* for(int i=0;i<total\_temp;i++) { burst[i]= burst[i]+ioint[]; }

         \*/

        burst[total\_temp / 2] = burst[total\_temp / 2] + ioint[total\_temp / 2];

        burst[total\_temp / 2 + 1] = burst[total\_temp / 2 + 1] + ioint[total\_temp / 2 + 1];

        float remain[] = new float[total\_temp];

        for (int i = 0; i < total\_temp; i++) {

            remain[i] = burst[i];

        }

        float exec = 0;

        float arrive = arrival[0];

        float quantum = 10;

        while (true) {

            boolean done = true;

            for (int i = 0; i < total\_temp; i++) {

                if (remain[i] > 0) {

                    done = false;

                    if (remain[i] > quantum && arrival[i] <= arrive) {

                        exec += quantum;

                        remain[i] -= quantum;

                        arrive++;

                    } else {

                        if (arrival[i] <= arrive) {

                            arrive++;

                            exec += remain[i];

                            remain[i] = 0;

                            exit[i] = exec;

                        }

                    }

                }

            }

            if (done == true) {

                break;

            }

        }

        for (int i = 0; i < total\_temp; i++) {

            turnaround[i] = exit[i] - arrival[i];

        }

        for (int i = 0; i < total\_temp; i++) {

            wait[i] = turnaround[i] - burst[i];

        }

        System.out.println("Turnaround " + " Waiting\t");

        float turn\_sum = 0;

        float wait\_sum = 0;

        for (int i = 0; i < total\_temp; i++) {

            turn\_sum += turnaround[i];

            wait\_sum += wait[i];

            System.out.println(" " + turnaround[i] + "\t\t " + wait[i]);

        }

        System.out.println("Average turn around time: " + turn\_sum / (float) total\_temp);

        System.out.println("Average waiting time: " + wait\_sum / (float) total\_temp);

    }

}

Output:-

