LABORATORY PRACTICE I

Practical Examination
SPPU AY 2021-22
Semester :- 1

NAME :- OJUS P. JAISWAL

YEAR & DIV :- TE A

ROLL NO. :- TACO19108

SEAT NO. :- S191094290

PRN NO. :- 72036776L

Assignment No. B7

Problem Statement :- Write a program to simulate CPU Scheduling Algorithms: FCFS, SJF (Preemptive), Priority (Non-Preemptive) and Round Robin (Preemptive).

Solution :-

Program :

a) FCFS =>

```
/* FCFS */

package B2;

import java.io.*;
import java.util.Scanner;
public class FCFS
{
	public static void main(String args[])
	{
		int i,no_p,burst_time[],TT[],WT[];
		float avg_wait=0,avg_TT=0;
		burst_time=new int[50];
		TT=new int[50];
		WT=new int[50];
		WT[0]=0;
		Scanner s=new Scanner(System.in);
		System.out.println("Enter the number of process: ");
		no_p=s.nextInt();
		System.out.println("\nEnter Burst Time for processes:");
		for(i=0;i<no_p;i++)
		{
			System.out.print("\tP"+(i+1)+":  ");
			burst_time[i]=s.nextInt();
		}

		for(i=1;i<no_p;i++)
		{
			WT[i]=WT[i-1]+burst_time[i-1];
			avg_wait+=WT[i];
		}
		avg_wait/=no_p;
```

```
                for(i=0;i<no_p;i++)
                {
                        TT[i]=WT[i]+burst_time[i];
                        avg_TT+=TT[i];
                }
                avg_TT/=no_p;

        System.out.println("\n**********************************************************
**********");
                System.out.println("\tProcesses:");

        System.out.println("**********************************************************
*********");
                System.out.println("    Process\tBurst Time\tWaiting Time\tTurn Around Time");
                for(i=0;i<no_p;i++)
                {
                        System.out.println("\tP"+(i+1)+"\t    "+burst_time[i]+"\t\t    "+WT[i]+"\t\t
"+TT[i]);

                }
                System.out.println("\n----------------------------------------------------------------");
                System.out.println("\nAverage waiting time : "+avg_wait);
                System.out.println("\nAverage Turn Around time : "+avg_TT+"\n");
        }
}
```

b) SJF (Preemptive) =>

```
/* SJF (Preemptive) */

package B2;

import java.util.*;

public class SJF {
public static void main (String args[])
{
Scanner sc=new Scanner(System.in);
System.out.println ("enter no of process:");
int n= sc.nextInt();
int pid[] = new int[n]; // it takes pid of process
int at[] = new int[n]; // at means arrival time
int bt[] = new int[n]; // bt means burst time
int ct[] = new int[n]; // ct means complete time
int ta[] = new int[n];// ta means turn around time
int wt[] = new int[n];  // wt means waiting time
int f[] = new int[n];  // f means it is flag it checks process is completed or not
int k[]= new int[n];   // it is also stores brust time
    int i, st=0, tot=0;
    float avgwt=0, avgta=0;

    for (i=0;i<n;i++)
    {
     pid[i]= i+1;
     System.out.println ("enter process " +(i+1)+ " arrival time:");
     at[i]= sc.nextInt();
     System.out.println("enter process " +(i+1)+ " burst time:");
     bt[i]= sc.nextInt();
     k[i]= bt[i];
     f[i]= 0;
    }

    while(true){
     int min=99,c=n;
     if (tot==n)
     break;

     for ( i=0;i<n;i++)
     {
     if ((at[i]<=st) && (f[i]==0) && (bt[i]<min))
     {
     min=bt[i];
```

```
        c=i;
        }
        }

        if (c==n)
        st++;
        else
        {
        bt[c]--;
        st++;
        if (bt[c]==0)
        {
        ct[c]= st;
        f[c]=1;
        tot++;
        }
        }
       }

       for(i=0;i<n;i++)
       {
        ta[i] = ct[i] - at[i];
        wt[i] = ta[i] - k[i];
        avgwt+= wt[i];
        avgta+= ta[i];
       }

       System.out.println("pid  arrival  burst  complete turn waiting");
       for(i=0;i<n;i++)
       {
        System.out.println(pid[i] +"\t"+ at[i]+"\t"+ k[i] +"\t"+ ct[i] +"\t"+ ta[i] +"\t"+ wt[i]);
       }

       System.out.println("\naverage tat is "+ (float)(avgta/n));
       System.out.println("average wt is "+ (float)(avgwt/n));
       sc.close();
}
}
```

c) Priority (Non-Preemptive) =>

```
/* Priority (Non-Preemptive) */

package B2;

import java.util.Scanner;

public class Priority
{

   int burstTime[];
   int priority[];
   int arrivalTime[];
   String[] processId;
   int numberOfProcess;

   void getProcessData(Scanner input)
   {
      System.out.print("Enter the number of Process for Scheduling          : ");
      int inputNumberOfProcess = input.nextInt();
      numberOfProcess = inputNumberOfProcess;
      burstTime = new int[numberOfProcess];
      priority = new int[numberOfProcess];
      arrivalTime = new int[numberOfProcess];
      processId = new String[numberOfProcess];
      String st = "P";
      for (int i = 0; i < numberOfProcess; i++)
      {
         processId[i] = st.concat(Integer.toString(i));
         System.out.print("Enter the burst time   for Process - " + (i) + " : ");
         burstTime[i] = input.nextInt();
         System.out.print("Enter the arrival time for Process - " + (i) + " : ");
         arrivalTime[i] = input.nextInt();
         System.out.print("Enter the priority     for Process - " + (i) + " : ");
         priority[i] = input.nextInt();
      }
   }

   void sortAccordingArrivalTimeAndPriority(int[] at, int[] bt, int[] prt, String[] pid)
   {

      int temp;
      String stemp;
      for (int i = 0; i < numberOfProcess; i++)
      {
```

```
for (int j = 0; j < numberOfProcess - i - 1; j++)
{
   if (at[j] > at[j + 1])
   {
      //swapping arrival time
      temp = at[j];
      at[j] = at[j + 1];
      at[j + 1] = temp;

      //swapping burst time
      temp = bt[j];
      bt[j] = bt[j + 1];
      bt[j + 1] = temp;

      //swapping priority
      temp = prt[j];
      prt[j] = prt[j + 1];
      prt[j + 1] = temp;

      //swapping process identity
      stemp = pid[j];
      pid[j] = pid[j + 1];
      pid[j + 1] = stemp;

   }
   //sorting according to priority when arrival timings are same
   if (at[j] == at[j + 1])
   {
      if (prt[j] > prt[j + 1])
      {
         //swapping arrival time
         temp = at[j];
         at[j] = at[j + 1];
         at[j + 1] = temp;

         //swapping burst time
         temp = bt[j];
         bt[j] = bt[j + 1];
         bt[j + 1] = temp;

         //swapping priority
         temp = prt[j];
         prt[j] = prt[j + 1];
         prt[j + 1] = temp;
```

```
                //swapping process identity
                stemp = pid[j];
                pid[j] = pid[j + 1];
                pid[j + 1] = stemp;

            }
        }
    }

  }
}

void priorityNonPreemptiveAlgorithm()
{
  int finishTime[] = new int[numberOfProcess];
  int bt[] = burstTime.clone();
  int at[] = arrivalTime.clone();
  int prt[] = priority.clone();
  String pid[] = processId.clone();
  int waitingTime[] = new int[numberOfProcess];
  int turnAroundTime[] = new int[numberOfProcess];

  sortAccordingArrivalTimeAndPriority(at, bt, prt, pid);

  //calculating waiting & turn-around time for each process
  finishTime[0] = at[0] + bt[0];
  turnAroundTime[0] = finishTime[0] - at[0];
  waitingTime[0] = turnAroundTime[0] - bt[0];

  for (int i = 1; i < numberOfProcess; i++)
  {
    finishTime[i] = bt[i] + finishTime[i - 1];
    turnAroundTime[i] = finishTime[i] - at[i];
    waitingTime[i] = turnAroundTime[i] - bt[i];
  }
  float sum = 0;
  for (int n : waitingTime)
  {
    sum += n;
  }
  float averageWaitingTime = sum / numberOfProcess;

  sum = 0;
  for (int n : turnAroundTime)
  {
    sum += n;
```

```
        }
        float averageTurnAroundTime = sum / numberOfProcess;

        //print on console the order of processes along with their finish time & turn around time
        System.out.println("Priority Scheduling Algorithm : ");
        System.out.format("%20s%20s%20s%20s%20s%20s\n",  "ProcessId",  "BurstTime",
"ArrivalTime", "Priority", "FinishTime", "WaitingTime", "TurnAroundTime");
        for (int i = 0; i < numberOfProcess; i++) {
           System.out.format("%20s%20d%20d%20d%20d%20d\n", pid[i], bt[i], at[i], prt[i],
finishTime[i], waitingTime[i], turnAroundTime[i]);
        }

        System.out.format("%100s%20f%20f\n",          "Average",          averageWaitingTime,
averageTurnAroundTime);
    }

    public static void main(String[] args)
    {
        Scanner input = new Scanner(System.in);
        Priority obj = new Priority();
        obj.getProcessData(input);
        obj.priorityNonPreemptiveAlgorithm();
    }
}
```

d) Round Robin (Preemptive) =>

```
/* Round Robin (Preemptive) */

package B2;

import java.util.*;

public class RoundRobin{
	private static Scanner inp = new Scanner(System.in);
	//Driver Code
	public static void main(String[] args){
		int n,tq, timer = 0, maxProccessIndex = 0;
		float avgWait = 0, avgTT = 0;
		System.out.print("\nEnter the time quanta : ");
		tq = inp.nextInt();
		System.out.print("\nEnter the number of processess : ");
		n = inp.nextInt();
		int arrival[] = new int[n];
		int burst[] = new int[n];
		int wait[] = new int[n];
		int turn[] = new int[n];
		int queue[] = new int[n];
		int temp_burst[] = new int[n];
		boolean complete[] = new boolean[n];

		System.out.print("\nEnter the arrival time of the processess : ");
		for(int i = 0; i < n; i++)
			arrival[i] = inp.nextInt();

		System.out.print("\nEnter the burst time of the processess : ");
		for(int i = 0; i < n; i++){
			burst[i] = inp.nextInt();
			temp_burst[i] = burst[i];
		}

		for(int i = 0; i < n; i++){ //Initializing the queue and complete array
			complete[i] = false;
			queue[i] = 0;
		}
		while(timer < arrival[0]) //Incrementing Timer until the first process arrives
			timer++;
		queue[0] = 1;

		while(true){
			boolean flag = true;
```

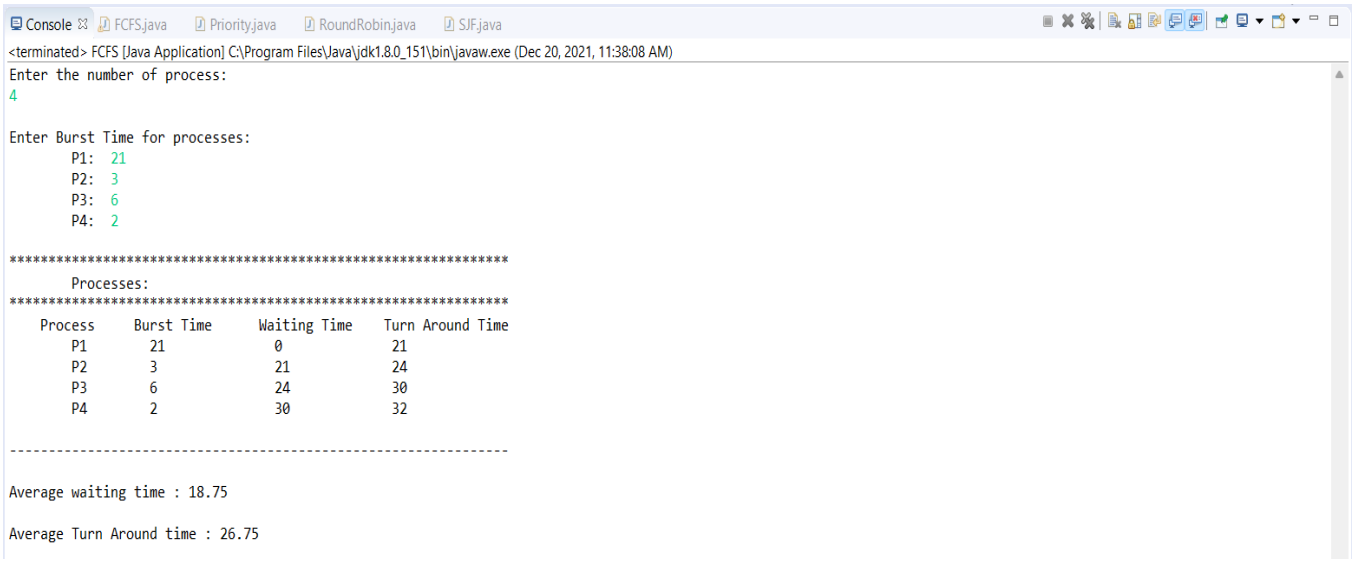
```
                for(int i = 0; i < n; i++){
                        if(temp_burst[i] != 0){
                                flag = false;
                                break;
                        }
                }
                if(flag)
                        break;

                for(int i = 0; (i < n) && (queue[i] != 0); i++){
                        int ctr = 0;
                        while((ctr < tq) && (temp_burst[queue[0]-1] > 0)){
                                temp_burst[queue[0]-1] -= 1;
                                timer += 1;
                                ctr++;

                                //Updating the ready queue until all the processes arrive
                                checkNewArrival(timer, arrival, n, maxProccessIndex,
queue);
                        }
                        if((temp_burst[queue[0]-1] == 0) && (complete[queue[0]-1] ==
false)){
                                turn[queue[0]-1] = timer;       //turn currently stores exit
times
                                complete[queue[0]-1] = true;
                        }

                        //checks whether or not CPU is idle
                        boolean idle = true;
                        if(queue[n-1] == 0){
                                for(int k = 0; k < n && queue[k] != 0; k++){
                                        if(complete[queue[k]-1] == false){
                                                idle = false;
                                        }
                                }
                        }
                        else
                                idle = false;

                        if(idle){
                                timer++;
                                checkNewArrival(timer, arrival, n, maxProccessIndex,
queue);
                        }
```

```
                        //Maintaining the entries of processes after each preemption in the
ready Queue
                        queueMaintainence(queue,n);
                    }
                }

                for(int i = 0; i < n; i++){
                    turn[i] = turn[i] - arrival[i];
                    wait[i] = turn[i] - burst[i];
                }

                System.out.print("\nProgram No.\tArrival Time\tBurst Time\tWait
Time\tTurnAround Time"
                                        + "\n");
                for(int i = 0; i < n; i++){
                    System.out.print(i+1+"\t\t"+arrival[i]+"\t\t"+burst[i]
                                            +"\t\t"+wait[i]+"\t\t"+turn[i]+ "\n");
                }
                for(int i =0; i< n; i++){
                    avgWait += wait[i];
                    avgTT += turn[i];
                }
                System.out.print("\nAverage wait time : "+(avgWait/n)
                                        +"\nAverage Turn Around Time : "+(avgTT/n));
        }
        public static void queueUpdation(int queue[],int timer,int arrival[],int n, int
maxProccessIndex){
                int zeroIndex = -1;
                for(int i = 0; i < n; i++){
                    if(queue[i] == 0){
                        zeroIndex = i;
                        break;
                    }
                }
                if(zeroIndex == -1)
                    return;
                queue[zeroIndex] = maxProccessIndex + 1;
        }

        public static void checkNewArrival(int timer, int arrival[], int n, int maxProccessIndex,int
queue[]){
                if(timer <= arrival[n-1]){
                    boolean newArrival = false;
                    for(int j = (maxProccessIndex+1); j < n; j++){
                        if(arrival[j] <= timer){
                            if(maxProccessIndex < j){
```

```
                                        maxProccessIndex = j;
                                        newArrival = true;
                                    }
                                }
                            }
                            if(newArrival) //adds the index of the arriving process(if any)
                                queueUpdation(queue,timer,arrival,n, maxProccessIndex);
                        }
                    }

                    public static void queueMaintainence(int queue[], int n){

                        for(int i = 0; (i < n-1) && (queue[i+1] != 0) ; i++){
                            int temp = queue[i];
                            queue[i] = queue[i+1];
                            queue[i+1] = temp;
                        }
                    }
                }
```

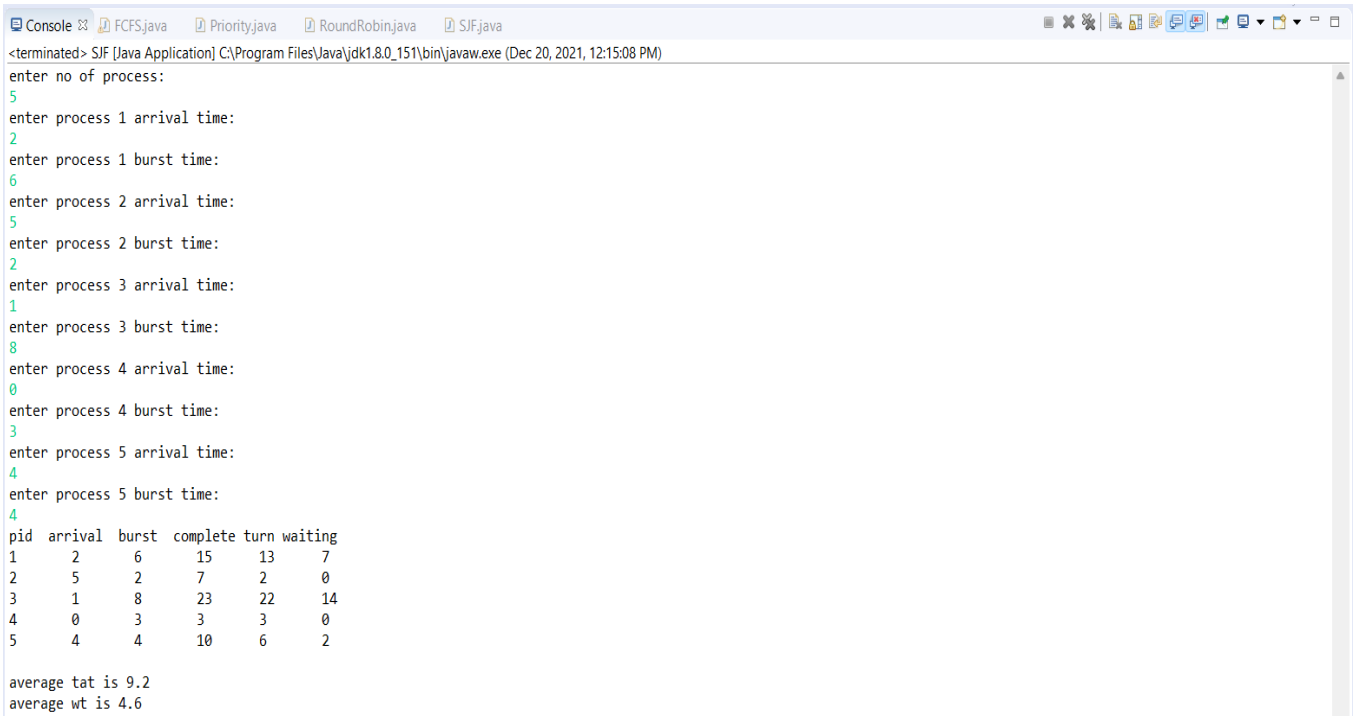
Output :

a) FCFS =>



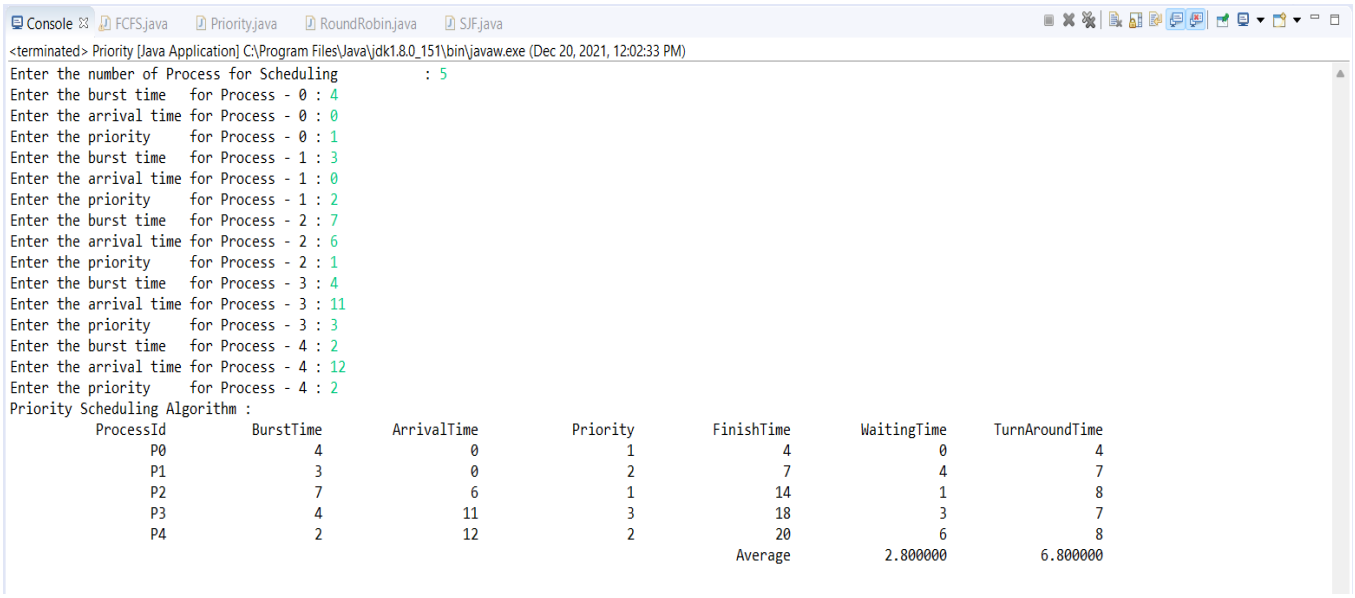
```
Console  FCFS.java  Priority.java  RoundRobin.java  SJF.java
<terminated> FCFS [Java Application] C:\Program Files\Java\jdk1.8.0_151\bin\javaw.exe (Dec 20, 2021, 11:38:08 AM)
Enter the number of process:
4

Enter Burst Time for processes:
        P1:  21
        P2:  3
        P3:  6
        P4:  2

*****************************************************************
        Processes:
*****************************************************************
    Process     Burst Time      Waiting Time    Turn Around Time
        P1        21              0               21
        P2        3               21              24
        P3        6               24              30
        P4        2               30              32

-----------------------------------------------------------------

Average waiting time : 18.75

Average Turn Around time : 26.75
```

b) SJF (Preemptive) =>



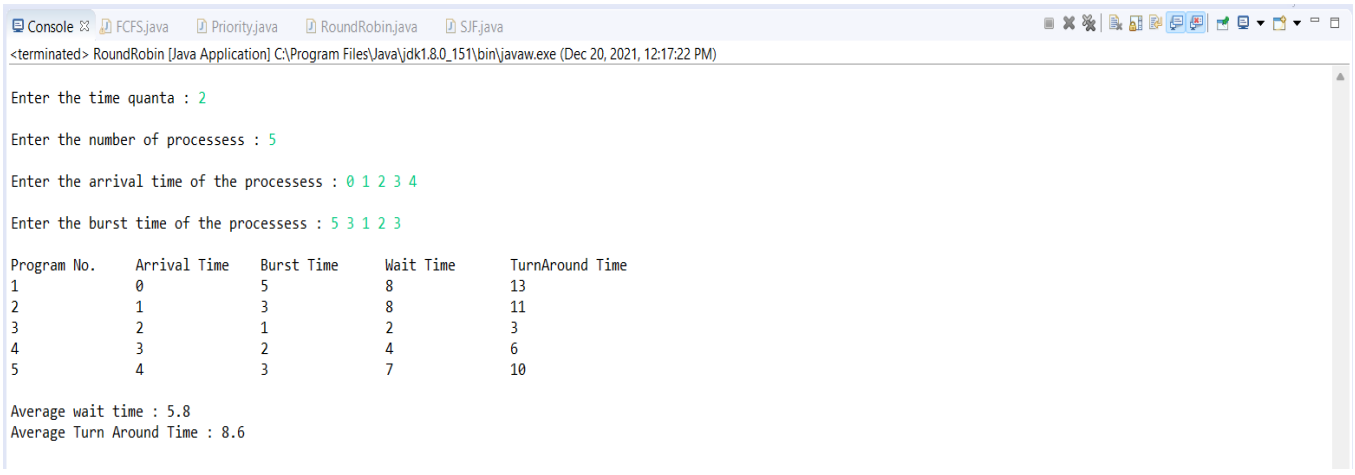
```
Console  FCFS.java  Priority.java  RoundRobin.java  SJF.java
<terminated> SJF [Java Application] C:\Program Files\Java\jdk1.8.0_151\bin\javaw.exe (Dec 20, 2021, 12:15:08 PM)
enter no of process:
5
enter process 1 arrival time:
2
enter process 1 burst time:
6
enter process 2 arrival time:
5
enter process 2 burst time:
2
enter process 3 arrival time:
1
enter process 3 burst time:
8
enter process 4 arrival time:
0
enter process 4 burst time:
3
enter process 5 arrival time:
4
enter process 5 burst time:
4
pid  arrival  burst  complete turn waiting
1       2       6       15      13      7
2       5       2       7       2       0
3       1       8       23      22      14
4       0       3       3       3       0
5       4       4       10      6       2

average tat is 9.2
average wt is 4.6
```

c) Priority (Non-Preemptive) =>



```
Console  FCFS.java  Priority.java  RoundRobin.java  SJF.java
<terminated> Priority [Java Application] C:\Program Files\Java\jdk1.8.0_151\bin\javaw.exe (Dec 20, 2021, 12:02:33 PM)
Enter the number of Process for Scheduling          : 5
Enter the burst time   for Process - 0 : 4
Enter the arrival time for Process - 0 : 0
Enter the priority     for Process - 0 : 1
Enter the burst time   for Process - 1 : 3
Enter the arrival time for Process - 1 : 0
Enter the priority     for Process - 1 : 2
Enter the burst time   for Process - 2 : 7
Enter the arrival time for Process - 2 : 6
Enter the priority     for Process - 2 : 1
Enter the burst time   for Process - 3 : 4
Enter the arrival time for Process - 3 : 11
Enter the priority     for Process - 3 : 3
Enter the burst time   for Process - 4 : 2
Enter the arrival time for Process - 4 : 12
Enter the priority     for Process - 4 : 2
Priority Scheduling Algorithm :
          ProcessId           BurstTime         ArrivalTime           Priority         FinishTime        WaitingTime      TurnAroundTime
                 P0                   4                   0                   1                   4                   0                   4
                 P1                   3                   0                   2                   7                   4                   7
                 P2                   7                   6                   1                  14                   1                   8
                 P3                   4                  11                   3                  18                   3                   7
                 P4                   2                  12                   2                  20                   6                   8
                                                                                              Average          2.800000          6.800000
```

d) Round Robin (Preemptive) =>



```
Console  FCFS.java  Priority.java  RoundRobin.java  SJF.java
<terminated> RoundRobin [Java Application] C:\Program Files\Java\jdk1.8.0_151\bin\javaw.exe (Dec 20, 2021, 12:17:22 PM)

Enter the time quanta : 2

Enter the number of processess : 5

Enter the arrival time of the processess : 0 1 2 3 4

Enter the burst time of the processess : 5 3 1 2 3

Program No.     Arrival Time    Burst Time      Wait Time       TurnAround Time
1               0               5               8               13
2               1               3               8               11
3               2               1               2               3
4               3               2               4               6
5               4               3               7               10

Average wait time : 5.8
Average Turn Around Time : 8.6
```