Grp
C

PAGE
DATE: / /

8th Assignment

Aim: Implement Job scheduling algorithm FCFS SJF Prieority Round Robin

Problem Statement : WAJP (using OOP features) to implement following algorithm
FCFS , SJF (Preemptive)
Prieority (Non Preemptive)
Round Robin (Preemptive)

Theory

① Problem Explanation

- CPU scheduling deals with the problem of deciding of which process in the ready queue is allowed to ~~otitize~~ utilize the CPU
- The criteria for selection for an algo are

① Maximum throughput
② least turnaround time
③ Minimum Waiting time
④ Maximum CPU utilization
⑤ Also the variance in response time must be minimum. In Preemptive job a currently

Executing job can be removed and a new job can take its place however in Non Preemptive this is not possible

First Come First Serve

(2) - Simplest CPU scheduling algo.

- the process that req the CPU 1st is the one to which it is allocated 1st
- the algo is implemented using a job queue
- when a process Req CPU it is added to the tail of job queue
- the CPU is allocated to the process at the head of the queue
- However the TAT varies which is not favourable
- Implementation

1 i/p process along with the burst time (bt)

2 Find waiting Time (wt) for all process.

3 As the 1st process that comes need no (wt)
so (wt) for 1st process = 0 ie wt[0]=0

4 Find the wt for all other process ie for
process i -> wt[i] = bt[i-1] + wt[i-1]

5 find turnaround time = wt + bt for all
processes.

6 Find avg waiting time = total_waiting_time/
no_of_process.

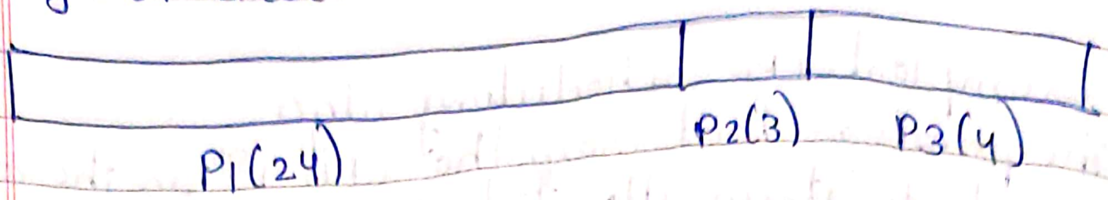
7 Similarly, find avg turn around time
total_turn_around_time/no_of_process.

Example

Process	Duration	Oder	Arrival Time
P1	24	1	0
P2	3	2	0
P3	4	3	0

PAGE:
DATE / /

Gantt chart



P1 (wt) = 0
P2 (wt) = 24
P3 (wt) = 27

avg wt = (0+24+27)/3
= 17

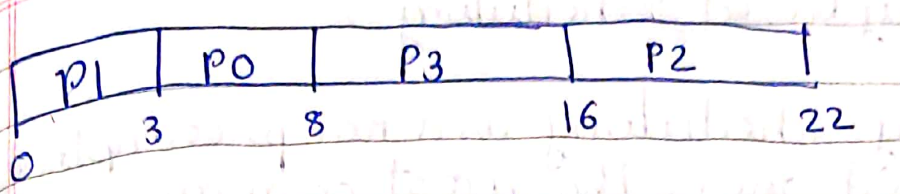
③ Shortest Job First

- This algo associates with its length of the next CPU burst
- when the CPU is available it is assigner to that job with smallest CPU burst.
- This algo provides min avg wtime
- The major problem with this knows the the CPU burst of a job.

3.1 Algorithm

1 Sort all the processes in increasing order according to burst time.
2 Then apply FCFS

Process	Arrival time	Execute Time	Servic
P0	0	5	0
P1	1	3	3
P2	2	8	8
P3	3	6	16



3.2 How to compute below time in SJF using a prog.

1 Completion time
Time at which process completes its execution

2 Turnaround time
Time difference between completion time & arrivel time

3 Waiting time
Time difference between Turn around time and burst time

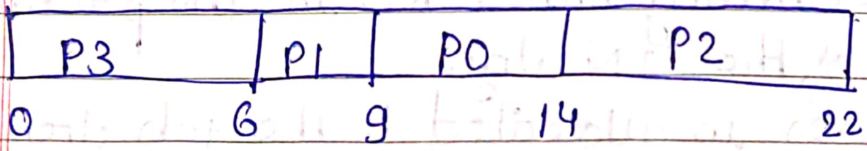
3.3 Shoertest Remaing Time

- Shoertest Remaing time (SRT) is the preemptiue verision of the SJN algo.
- The processes is allocated to the job closest to completion but it can be preempted by a newer ready job with shoertes time to completion
- Impossible to implement in interactive system where required CPU time is not known.
- It is often used in batch enuironment where short job need to give preference

(4) Priority Based scheduling

- Priority scheduling is a non preemptive algo and one of the most common scheduling algo in batch system.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Process with same priority are executed on FCFS basis.
- Priority can be decided based on memory requirements or any other resource requirement

Process	Arrival time	Execute time	Priority	time Servi
P0	0	5	1	9
P1	1	3	2	6
P2	2	8	1	14
P3	3	6	3	0



4.1 Implementation

1 1st i/p the process with burst time and priority

2 Sort the process, bt and priority according to priority

3 Now simply apply FCFS algo

- Note : A major problem with priority scheduling is indefinite blocking or starvation.
- A solution to the problem of indefinite blockage of the low priority process is aging
- Aging is a technique of gradually increasing priority of process that wait in the system for a long period of time

wt for each process is as follows

Process	wt = St - At
P0	9-0 = 9
P1	6-1 = 5
P2	14-2 = 12
P3	0-0 = 0.

⑤ Round Robin

- It is a CPU scheduling where each process is assigned a fixed time slot in a cyclic way
- It is a simple, easy to implement and starvation free as all process get fair share of CPU
- One of the most commonly used Technique in CPU scheduling as a core.
- It is preemptive as processes are assigned CPU only for a fixed slice of a time at most
- The disadvantage of it more overhead of context switching

- Each process is provided a fix time to execu it called quantum
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processes.

Quantum 3

P0	P1	P2	P3	P0	P2	P3	P2

0 3 6 9 12 14 17 20 22

Example

Process	Duration	Order	Arival time
P1	3	1	0
P2	4	2	0
P3	3	3	0

Quantum 1

P1	P2	P3	P1	P2	P3	P1	P2	P3	P2

0 10

P1 wt = 4 avg wt =(4+6+6)/3 = 5.33

P2 wt = 6

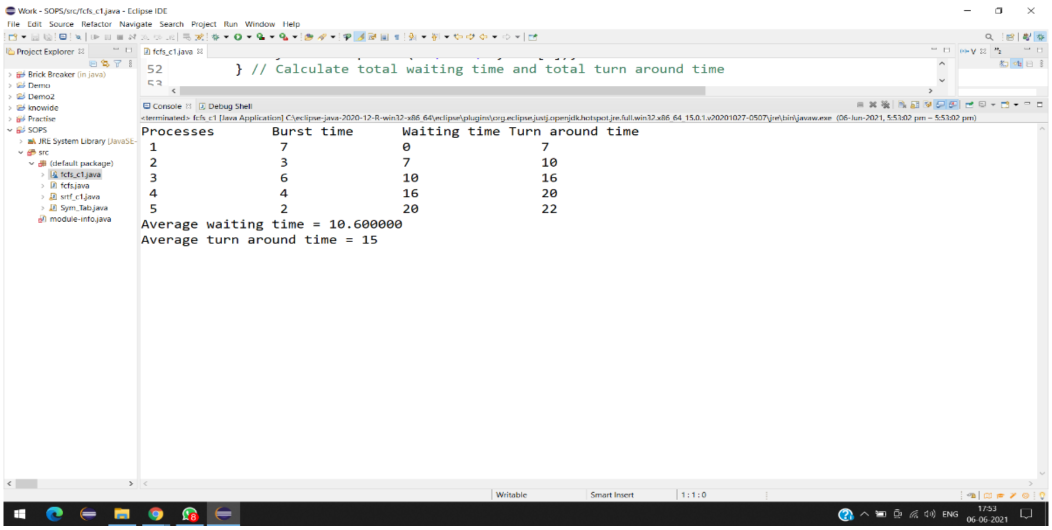
P3 wt = 6

Conclusion Thus we have implemented FCFS, SJF, Priority and Round Robin algorithm.

*******************FCFS**************************

```
//Name Ankita Bonde
// TE-A 19
//  ASSINGNMENT:GROUP_C_1
//Java program for implementation of FCFS
// scheduling
import java.util.*;

public class srtf_c1 {
        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();
    }
}
```

_______________OUTPUT________________



```
        } // Calculate total waiting time and total turn around time
```

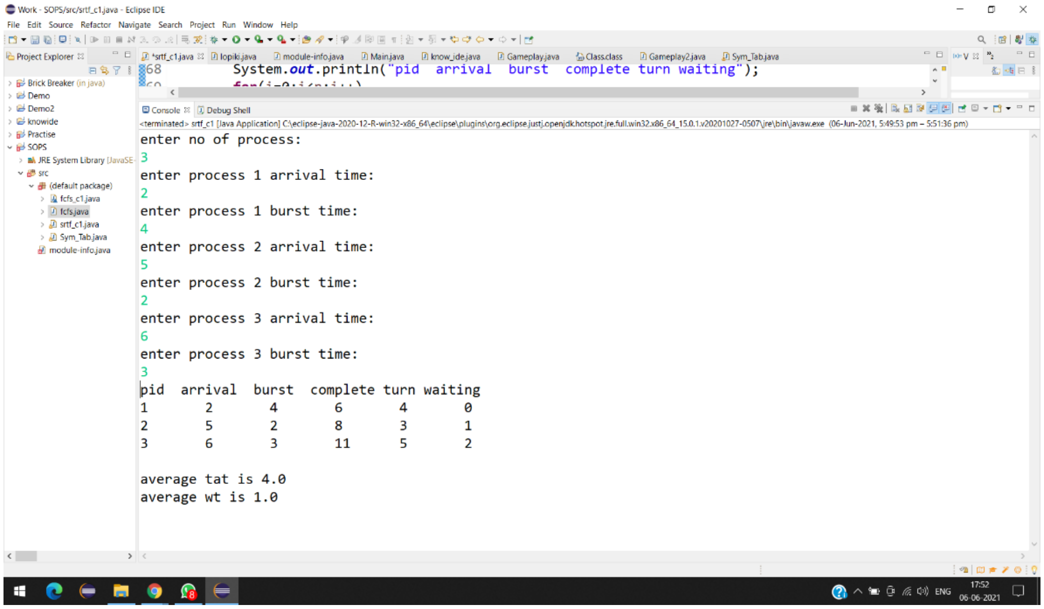
<terminated> fcfs_c1 [Java Application] C:\eclipse-java-2020-12-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.1.v20201027-0507\jre\bin\javaw.exe (06-Jun-2021, 5:53:02 pm – 5:53:02 pm)

```
Processes        Burst time       Waiting time Turn around time
 1                7               0                 7
 2                3               7                 10
 3                6               10                16
 4                4               16                20
 5                2               20                22
Average waiting time = 10.600000
Average turn around time = 15
```

********************SRTF***************************

```
import java.util.*;
public class srtf_c1 {
        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();}}
```

________________OUTPUT___________________________



```
        System.out.println("pid  arrival  burst  complete turn waiting");
```

```
<terminated> srtf_c1 [Java Application] C:\eclipse-java-2020-12-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.1.v20201027-0507\jre\bin\javaw.exe (06-Jun-2021, 5:49:53 pm – 5:51:36 pm)
enter no of process:
3
enter process 1 arrival time:
2
enter process 1 burst time:
4
enter process 2 arrival time:
5
enter process 2 burst time:
2
enter process 3 arrival time:
6
enter process 3 burst time:
3
pid  arrival  burst  complete turn waiting
1       2       4       6       4       0
2       5       2       8       3       1
3       6       3       11      5       2


average tat is 4.0
average wt is 1.0
```

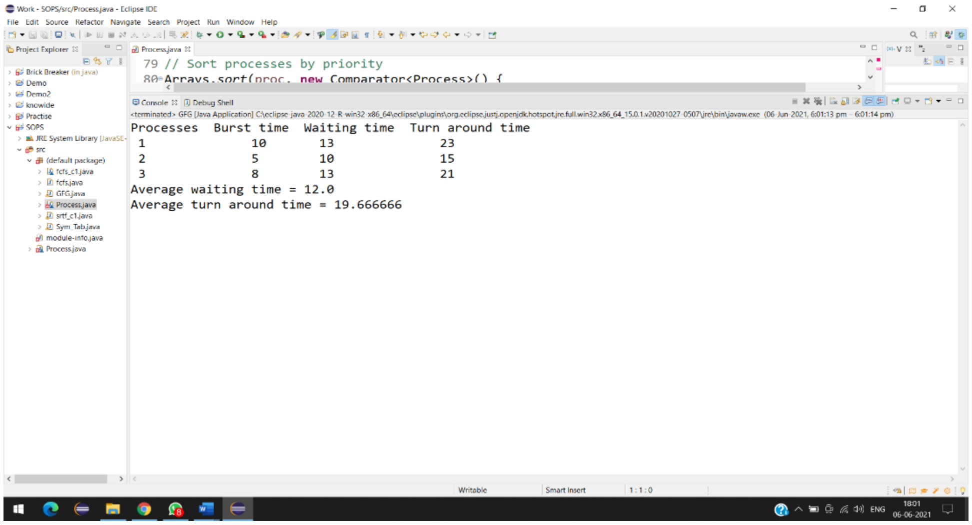
**********************PRIORITY*****************************

```
import java.util.*;

class Process
{
int pid; // Process ID
int bt; // CPU Burst time required
int priority; // Priority of this process
Process(int pid, int bt, int priority)
{
this.pid = pid;
this.bt = bt;
this.priority = priority;
}
public int prior() {
return priority;
}
}


public class GFG
{
```

```
// Function to find the waiting time for all
// processes
public void findWaitingTime(Process proc[], int n,
int wt[])
{

// waiting time for first process is 0
wt[0] = 0;

// calculating waiting time
for (int i = 1; i < n ; i++ )
wt[i] = proc[i - 1].bt + wt[i - 1] ;
}

// Function to calculate turn around time
public void findTurnAroundTime( Process proc[], int n,
int wt[], int tat[])
{
// calculating turnaround time by adding
// bt[i] + wt[i]
for (int i = 0; i < n ; i++)
tat[i] = proc[i].bt + wt[i];
}

// Function to calculate average time
public void findavgTime(Process proc[], int n)
{
int wt[] = new int[n], tat[] = new int[n], total_wt = 0, total_tat = 0;

// Function to find waiting time of all processes
findWaitingTime(proc, n, wt);

// Function to find turn around time for all processes
findTurnAroundTime(proc, n, wt, tat);

// Display processes along with all details
System.out.print("\nProcesses   Burst time   Waiting time   Turn around time\n");

// Calculate total waiting time and total turn
// around time
for (int i = 0; i < n; i++)
{
total_wt = total_wt + wt[i];
total_tat = total_tat + tat[i];
System.out.print(" " + proc[i].pid + "\t\t" + proc[i].bt + "\t " + wt[i] + "\t\t " + tat[i] + "\n");
}

System.out.print("\nAverage waiting time = "
+(float)total_wt / (float)n);
System.out.print("\nAverage turn around time = "+(float)total_tat / (float)n);
}
```

```
public void priorityScheduling(Process proc[], int n)
{

// Sort processes by priority
Arrays.sort(proc, new Comparator<Process>() {
@Override
public int compare(Process a, Process b) {
return b.prior() - a.prior();
}
});
System.out.print("Order in which processes gets executed \n");
for (int i = 0 ; i < n; i++)
System.out.print(proc[i].pid + " ") ;

findavgTime(proc, n);
}

// Driver code
public static void main(String[] args)
{
GFG ob=new GFG();
int n = 3;
Process proc[] = new Process[n];
proc[0] = new Process(1, 10, 2);
proc[1] = new Process(2, 5, 0);
proc[2] = new Process(3, 8, 1);
ob.priorityScheduling(proc, n);
}
}
```

_______________OUTPUT________________________



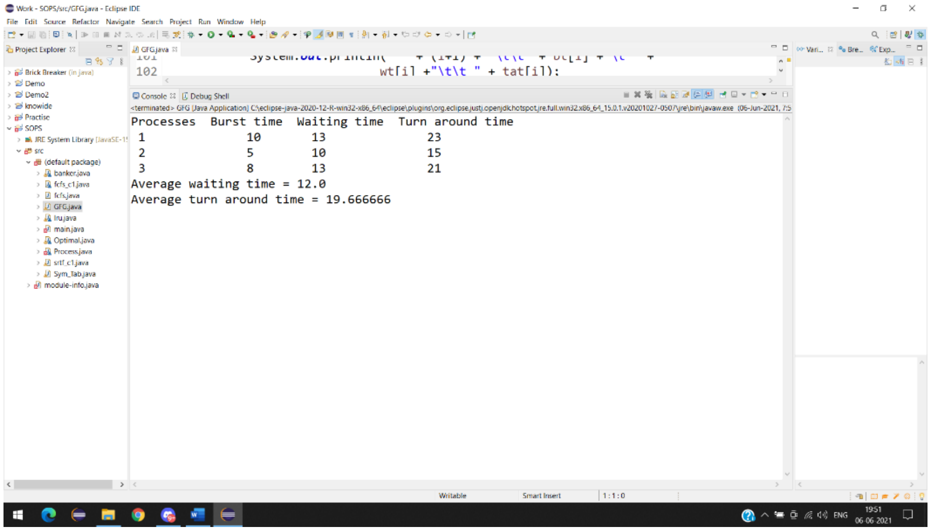
```
// Sort processes by priority
Arrays.sort(proc, new Comparator<Process>() {
```

```
<terminated> GFG [Java Application] C:\eclipse-java-2020-12-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.1.v20201027-0507\jre\bin\javaw.exe  (06-Jun-2021, 6:01:13 pm – 6:01:14 pm)
Processes  Burst time  Waiting time  Turn around time
 1            10         13              23
 2            5          10              15
 3            8          13              21
Average waiting time = 12.0
Average turn around time = 19.666666
```

**********************ROUND ROBIN*****************************

```
public class GFG
{
    static void findWaitingTime(int processes[], int n,
                 int bt[], int wt[], int quantum)
    {
        // Make a copy of burst times bt[] to store remaining
        // burst times.
        int rem_bt[] = new int[n];
        for (int i = 0 ; i < n ; i++)
            rem_bt[i] =  bt[i];

        int t = 0; // Current time


        while(true)
        {
            boolean done = true;
            for (int i = 0 ; i < n; i++)
            {
                if (rem_bt[i] > 0)
                {
                    done = false; // There is a pending process

                    if (rem_bt[i] > quantum)
                    {
                        t += quantum;
                       rem_bt[i] -= quantum;
                    }
                    else
                    {
                        t = t + rem_bt[i];
                       wt[i] = t - bt[i];
                        rem_bt[i] = 0;
                    }
                }
            }
            if (done == true)
              break;
        }
    }

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

        findWaitingTime(processes, n, bt, wt, quantum);
        findTurnAroundTime(processes, n, bt, wt, tat);
       System.out.println("Processes " + " Burst time " +
                      " Waiting time " + " Turn 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) + "\t\t" + bt[i] +"\t " +
                              wt[i] +"\t\t " + tat[i]);
        }

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

    public static void main(String[] args)
    {

        int processes[] = { 1, 2, 3};
        int n = processes.length;

        int burst_time[] = {10, 5, 8};

        int quantum = 2;
        findavgTime(processes, n, burst_time, quantum);
    }
}
```

OUTPUT



```
                                        wt[i] +"\t\t " + tat[i]);
```

<terminated> GFG [Java Application] C:\eclipse-java-2020-12-R-win32-x86_64\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_15.0.1.v20201027-0507\jre\bin\javaw.exe (06-Jun-2021, 7:5

```
Processes  Burst time  Waiting time  Turn around time
 1              10       13              23
 2              5        10              15
 3              8        13              21
Average waiting time = 12.0
Average turn around time = 19.666666
```