Name of the Experiment :

Page No.: 1
Date:
Practical No.: 1

Aim :

To write a Java program to stimulate the CPU scheduling algorithm first come first serve (FCFS)

Description :

TO calculate the average waiting time using the FCFS algorithm first the waiting time of the first process is kept zero and the waiting time of the second process is the burst time of the first process and the waiting time of the third process is the sum of the burst time of the first and the second process and so on. After calculating all the waiting times the average waiting is calculated as the average of all the waiting times. FCFS mainly says first come first serve the algorithm which came first will be served first

Algorithm :

step 1 : start the process

step 2 : Accept the number of process in the ready Queue

step 3 : for each process in the ready Q, assign the process name and the burst time step

step 4 : set the waiting of the first process as '0' and its burst time as its turnaround time step

step 5 : for each process in the ready Q calculate

a. Turnaround time = waiting time (n) + Burst time (n)
(n)

b. waiting time (n) = waiting time (n-1) + Burst time (n-1)

step 6 : calculate

a) Average waiting time = $\frac{\text{total waiting time}}{\text{Number of Process}}$

b. Average turnd around time = $\frac{\text{Total Turnaround time}}{\text{Number of Process}}$

Step 7 : Stop the process.

Source code :

```
import java.util.*;
public class FCFS {
    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]; // process ids
    int ar[] = new int[n]; // arrival times
    int bt[] = new int[n]; // burst time
    int ct[] = new int [n]; // completion time
    int ta[] = new int[n]; // turn around time
    int wt[] = new int[n]; // waiting time

    int temp;
    float avgwt = 0, avgta = 0;
    for (int i=0; i<n; i++)
    {
```

```
System.out.println("enter process" + (i+1) +
                   " arrival time :");
ar[i] = sc.nextInt();
System.out.println("enter process" + (i+1) +
                   " burst time :");
bt[i] = sc.nextInt();
pid[i] = i+1;
}
// sorting according to arrival times
for (int i=0; i<n; i++)
{
  for (int j=0; j<n-(i+1); j++)
  {
    if(ar[j] > ar[j+1])
    {
      temp = ar[j];
      ar[j] = ar[j+1];
      ar[j+1] = temp;
```

```
            temp = bt[j];
            bt[j] = bt[j+1];
            bt[j+1] = temp;
            temp = Pid[j];
            Pid[j] = Pid[j+1];
            Pid[j+1] = temp;
            }
        }
}
// finding completion times
for(int i=0; i<n; i++)
{
   if (i==0)
   {
     ct[i] = ar[i] + bt[i];
   }
   else
   {
     if (ar[i] > ct[i-1])
```

```
    {
      ct[i] = ar[i] + bt[i];
    }
    else
      ct[i] = ct[i-1] + bt[i];
  }
  ta[i] = ct[i] - ar[i]; // turnaround time = completion time -
                           arrival time
  wt[i] = ta[i] - bt[i]; // waiting time = turnaround time -
                           burst time
  avg wt += wt[i]; // total waiting time
  avg ta += ta[i]; // total turnaround time
}
System.out.println("\nPid arrival burst complete turn
                    waiting");
for (int i=0; i<n; i++)
{
  System.out.println(pid[i]+"\t"+ar[i]+"\t"+
      bt[i]+"\t"+ct[i]+"\t"+ta[i]+"\t"+wt[i]);
```

```
}
sc.close();
System.out.println("\naverage waiting time:" +
                   (avgwt/n));
System.out.println("average turnaround time:" +
                   (avgta/n));
}
}
```

Output :

PID	Arrival time	Burst time
P1	0	24
P2	1	3
P3	2	3

PID	Arrival time	Burst time	completim time	turnaround time	waiting time
P1	0	24	24	24-0=24	24-24=0
P2	1	3	27	27-1=26	26-3=23
P3	2	3	30	30-2=28	28-3=25

Average Turn around time = $\frac{24+26+28}{3}$ = 26ms

Average waiting time = $\frac{23+25}{3}$ = 16ms

Page No.: 10
Date:
Name of the Experiment :
Practical No.: 2

Aim :

To write a Java program to stimulate the CPU scheduling algorithm Shortest Job Next (SJN)

Description :

To calculate the average waiting time using the SJN algorithm first the waiting time of the first processor is kept zero and the waiting time of the second process is the burst time of the first process and the waiting time of the third process is the sum of the burst times of the first and the second process and so on. After calculating all the waiting times the average waiting time is calculated as the average of all waiting times. SJN mainly says that which one is the shortest job that will serve first.

Name of the Experiment :

Page No.: 11
Date:
Practical No.:

Algorithm :-

Step 1 : Start the process

Step 2 : Accept the number of processes in the ready Queue

Step 3 : for each process in the ready Q, assign the process name and the burst time.

Step 4 : Set the waiting of the first process as '0' and its burst time as its turn around time

Step 5 : for each process in the Ready Q calculate

a) waiting time (n) = waiting time (n-1) + Burst time (n-1)

b) Turn around time (n) = waiting time (n) + Burst time (n)

Step 6 : Calculate

a) Average Waiting time = $\frac{\text{total waiting time}}{\text{Number of process.}}$

b. Average turn around time = $\frac{\text{Total turnaround time}}{\text{Number of process}}$

Step 7 : Stop the process.

Name of the Experiment :

Source code :

```
import java.io.*
import java.util.*;
Public class Maain{
  Public Static void main (String[] args)
  {
   Scanner input =new Scanner (System.in);
   int n;
   //Matrix for storing Process id, Burst time,
   // Average waiting Time & Average turn around
   // time
   int[][]A = new int [100][4];
   int total =0;
   float avg_wt, avg_tat;
   System.out.println (" Enter no. of Process :");
   n = input.nextInt();
   System.out.println (" Enter Burst time: ");
   for (int i=0 ; i<n ; i++) {
   //user input Burst Time and alloting
   // ProcessId
```

```
        System.out.println("P" + (i+1) + " :");
        A[i][1] = input.nextInt();
        A[i][0] = i+1;

    }
    for (int i=0; i<n; i++) {
    // sorting process according to their Burst time
    int index i;
    for (int j=i+1; j<n; j++) {
      if (A[j][1] < A[index][1]) {
      index = j;
      }

}
int temp = A[i][1];
A[i][1] = A[index][1];
A[index][1] = temp;
temp = A[i][0];
A[i][0] = A[index][0];
A[index][0] = temp;

}
```

```
A[0][2] =0;
// calculation of waiting times
for (int i=1; i<n; i++){
  A[i][2] =0;
 for (int j=0; j<i; j++){
  A[i][2] += A[j][1];
 }
 total += A[i][2];
}
avg_wt = (float) total /n;
total=0;
// calculation of turnaround time and printing the data.
System.out.println("P\tBT\tWT\tTAT");
for(int i=0; i<n; i++){
  A[i][3] = A[i][1] + A[i][2];
  total+ = A[i][3];
 System.Out.println("P" + A[i][0]+"\t" + A[i][1] +
                    "\t" + A[i][2] + "\t" + A[i][3]
}
```

```
avg_tat = (float) total / n;
System.out.println("Average waiting time = " +
                    avg_wt);
System.out.println("Average Turnaround Time = " +
                    avg_tat);
    }
}
```

Output :

processors	Burst time
P1	8
P2	3
P3	9
P4	6

processors	Burst time	waiting time	Turn around time
P1	8	0	8
P2	3	3	6
P3	9	18	27
P4	6	18	24

Average waiting time = 9.75

Average turnaround time = 16.25

Name of the Experiment :

Aim :

To write a Java program to stimulate the CPU scheduling algorithm Priority Based

Description :

To calculate the average waiting time using the priority Based algorithm first the waiting time of the first process is kept zero and the waiting time of the second process is the burst time of the first process and the waiting time of the third process is the sum of the burst time of the first and the second process and so on. After calculating all the waiting time the average waiting time is calculated as the average of all the waiting times. Priority Based algorithms says that which we give the first priority that comes first as result.

Name of the Experiment :

Page No.: 19
Date:
Practical No.:

Algorithm :

step 1 : Start the process

step 2 : Accept the number of process in the ready Queue

step 3 : for each process in the ready Q, assign the process name and the burst time.

step 4 : set the waiting of the first process as '0' and its burst time as its turn around time.

step 5 : for each process in the ready Q calculate

a. waiting time(n) = waiting time(n-1) + Burst time (n-1)

b. Turn around time (n) = waiting time (n) + Burst time(n)

step 6 : calculate

a. Average waiting time = Total waiting time / Number of Process

ame of the Experiment :

b. Average Turnaround time $= \frac{\text{Total turn around time}}{\text{Number of process}}$

Step 7 : Stop the process

Source code :

```
class process {
    constructor ( Pid , bt , Priority ) {
      this . Pid = Pid ; // Process ID
      this . bt = bt ; // CPU Burst time required
      this . priority = Priority ; // Priority of this process
  }

  prior ( ) {
      return this . Priority ;
    }
}

class GFG {
    // function to find the waiting time for all
    // Processes
  findWaitingTime ( Proc , n , wt ) {
    // waiting time for first process is 0
    wt [0] = 0 ;
```

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

}

// function to calculate turn around time
findTurnAroundTime (proc, n, wt, tat) {
// calculating turnaround time by adding bt[i] + wt[i]
for (let i=0; i<n; i++){
 tat[i] = proc[i] bt + wt[i];
}

}

// function to calculate average time
findavgTime (proc, n) {
  let wt = new Array (n);
  let tat = new Array (n);
  let total_wt = 0;
```

```
// function to find waiting time of all processes
this.findWaitingTime(proc, n, wt);

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

// Display processsess along with all details
console.log(" Process ess  Burst time  waiting time
                Turn around time");

// calculate total waiting time and total turn around
// time
for(let i=0; i<n; i++){
    total_wt = total_wt + wt[i];
    total_tat = total_tat + tat[i];
    console.log("  "+proc[i].Pid+"\t\t"+proc[i].bt+
                "\t"+wt[i]+"\t\t"+tat[i]);
}
console.log("Average waiting time ="+total_wt/n);
```

```
console.log("Average turnaround time = " + total_tat /n);
}
PriorityScheduling (proc, n){
  //sort processes by priority
  proc.sort((a,b) => b.prior() - a.prior());
  console.log(" order in which processes get executed :");
  for (let i=0; i<n; i++){
  console.log(proc[i].Pid + "   ");
  }
   this.findavgTime (proc, n);
  }
}
// Driver code
let ob = new GFG();
let n = 3;
let proc = [];
 proc[0] = new Process(1, 10, 2);
 proc[1] = new Process (2, 5, 0);
```

Page No.: 25

Date:

Name of the Experiment :

Practical No.:

```
Proc[2] = new Process (3,8,1);
Ob. priority Scheduling (Proc,n);
```

Output :

Order in which processes gets executed

1 3 2

Processes	Burst time	waiting time	turnaround time
1	10	0	10
3	8	10	18
2	5	18	23

Average waiting time = 9.33333

Average turnaround time = 17

Name of the Experiment :

Page No.: 27
Date:
Practical No.: 4

Aim :

To write a Java program to stimulate the CPU scheduling algorithm round robin

Description :

To calculate the average waiting time using the round robin algorithm first the waiting time of the first process is kept zero and the waiting time of the second process is the burst time of the first process and the waiting time of the third process is the sum of the burst time of the first and the second process and soon. After calculating all the waiting times the average waiting time is calculated as the average of all waiting times.

Page No.: 28
Date:
Practical No.:
Name of the Experiment :

Algorithm :

Step 1 : Start the process

Step 2 : Accept the number of process in the ready Queue

Step 3 : for each process in the ready Q, assign the process name and the burst time

Step 4 : set the waiting of the first process as '0' and its burst time as its turn around time.

Step 5 : for each process in the Ready Q calculate

a. waiting time (n) = waiting time (n-1) + Burst time (n-1)

b. turnaround time (n) = waiting time (n) + Burst time (n).

Step 6 : calculate

a. Average waiting time = $\frac{\text{total waiting time}}{\text{Number of Process}}$

b. Turn around time = $\frac{\text{Total turn around time}}{\text{Number of process}}$

Step 7 : Stop the process.

Name of the Experiment :

Source Code :

```
Public class GFG{
  // method to find the waiting time for all processes
Static void find waitingtime (int processes[], intn, int bt[],
                              intwt[], 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
 // keep traversing processes in round robin manner
 // util all of them are not done
 while (true) {
    boolean done = true;
 // traverse all processes one by one repeatedly
 for (int i=0; i<n; i++) {
```

```
// if burst time of a process is greater than 0
//then only need to process further
if (rem_bt[i] >0) {
done = false; // There is a pending process
if (rem_bt[i] >quantum) {
  //increase the value of t i.e. shows how much time
  // process has been processed
  t += quantum;
}
// If burst time is smaller than or equal to 0
// quantum. last cycle for this process
else {
// increase the value of t i.e. shows how much time
// a process has been processed
t = t + rem_bt[i];
// waiting time is current time minus time used by
// this process
```

```
            wt[i] = t - bt[i];
        // As process gets fully executed makes its remaining
        // Burst time = 0
        rem_bt[i] = 0;
            }
        }
    }
    // If all processes are done
    if( done == true)
        break;
    }
}
// method to calculate turn around time
Static void findturnaroundtime( int processes[],
                                int n, int bt[], int wt[],
                                int tat[])
{
```

```
// calculating turnaround time by adding
// bt[i] + wt[i]
for (int i=0; i<n; i++)
  tat[i] = bt[i] + wt[i];
}
// method to calculate average time
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;
// function to find waiting time of all processes
findwaitingTime(processes, n, bt, wt, quantum);
// function to find turn around time for all Processes
findTurnaroundTime(processes, n, bt, wt, tat);
// display processes along with all details
```

```
System.out.println(" PN" + "B" + "WT " + "TAT");
// calculate total waiting time and total turnaround
//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" + 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);
}
// driver method
```

```
Public static void main(String[] args)
{
 // process id's
 int processes[] = {1,2,3};
 int n = processes.length;
 // Burst time of all processes
 int burst_time[] = {10,5,8};
// Time quantum
 int quantum = 2;
 findavgTime(processes, n, burst_time, quantum);
}
}
```

Page No.: 36

Date:

Practical No.:

lame of the Experiment :

Output :

PN	BT	WT	TAT
1	10	13	23
2	5	10	15
3	8	13	21

Average waiting time = 12

Average turn around time = 19.6667