Assignment 1

//program of pass1 of 2 pass assembler

import java.io.\*;

class sym {

public static void main(String arg[])throws IOException

{

//BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

int i;

String a[][]={

{"","START","101",""},

{"","MOVER","BREG","ONE"},

{"AGAIN","MULT","BREG","TERM"},

{"","MOVER","CREG","TERM"},

{"","ADD","CREG","N"},

{"","MOVEM","CREG","TERM"},

{"N","DS","2",""},

{"RESULT","DS","2",""},

{"ONE","DC","1",""},

{"TERM","DS","1",""},

{"","END","",""}};

int lc=Integer.parseInt(a[0][2]);//return int value

String st[][]=new String[5][2];//store symbol table

int cnt=0,l;

for (i=1;i<11;i++)

{

if (a[i][0]!="")

{

st [cnt][0]=a[i][0];

st[cnt][1]=Integer.toString(lc);

cnt++;

if(a[i][1]=="DS")

{

int d=Integer.parseInt(a[i][2]);

lc=lc+d;

}

else

{

lc++;

}

}

else

{

lc++;

}

}

System.out.print("symbol table");

for(i=0;i<5;i++)

{

for(cnt=0;cnt<2;cnt++)

{

System.out.print(st[i][cnt]+"\t");

}

System.out.println();

}

}

}

Assignment 3

FCFS program

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 or execution times

int ct[] = new int[n];     // completion times

int ta[] = new int[n];     // turn around times

int wt[] = new int[n];     // waiting times

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) + " brust 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

avgwt += wt[i] ;               // total waiting time

avgta += ta[i] ;               // total turnaround time

}

System.out.println("\npid  arrival  brust  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));     // printing average waiting time.

System.out.println("average turnaround time:"+(avgta/n));    // printing average turnaround time.

}

}

SJF(Preemptive)

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];

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 st=0, tot=0;

float avgwt=0, avgta=0;

for(int i=0;i<n;i++)

{

System.out.println ("enter process " + (i+1) + " arrival time:");

at[i] = sc.nextInt();

System.out.println ("enter process " + (i+1) + " brust time:");

bt[i] = sc.nextInt();

pid[i] = i+1;

f[i] = 0;

}

boolean a = true;

while(true)

{

int c=n, min=999;

if (tot == n) // total no of process = completed process loop will be terminated

break;

for (int i=0; i<n; i++)

{

/\*

\* If i'th process arrival time <= system time and its flag=0 and burst<min

\* That process will be executed first

\*/

if ((at[i] <= st) && (f[i] == 0) && (bt[i]<min))

{

min=bt[i];

c=i;

}

}

/\* If c==n means c value can not updated because no process arrival time< system time so we increase the system time \*/

if (c==n)

st++;

else

{

ct[c]=st+bt[c];

st+=bt[c];

ta[c]=ct[c]-at[c];

wt[c]=ta[c]-bt[c];

f[c]=1;

tot++;

}

}

System.out.println("\npid  arrival brust  complete turn waiting");

for(int i=0;i<n;i++)

{

avgwt+= wt[i];

avgta+= ta[i];

System.out.println(pid[i]+"\t"+at[i]+"\t"+bt[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();

}

}

Priority (non-preemptive)

import java.util.Arrays;

import java.util.Scanner;

public class Priority {

public static void main(String[] args) {

System.out.println("\*\*\* Priority Scheduling \*\*\*");

System.out.print("Enter Number of Process: ");

Scanner sc = new Scanner(System.in);

int numberOfProcess = sc.nextInt();

String process[] = new String[numberOfProcess];

int p = 1;

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

process[i] = "P" + p;

p++;

}

System.out.println(Arrays.toString(process));

System.out.print("Enter Burst Time for " + numberOfProcess + " process: ");

int burstTime[] = new int[numberOfProcess];

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

burstTime[i] = sc.nextInt();

}

System.out.println(Arrays.toString(burstTime));

System.out.print("Enter Priority for " + numberOfProcess + " process: ");

int priority[] = new int[numberOfProcess];

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

priority[i] = sc.nextInt();

}

System.out.println(Arrays.toString(priority));

// Sorting process & burst time by priority

int temp;

String temp2;

for (int i = 0; i < numberOfProcess - 1; i++) {

for (int j = 0; j < numberOfProcess - 1; j++) {

if (priority[j] > priority[j + 1]) {

temp = priority[j];

priority[j] = priority[j + 1];

priority[j + 1] = temp;

temp = burstTime[j];

burstTime[j] = burstTime[j + 1];

burstTime[j + 1] = temp;

temp2 = process[j];

process[j] = process[j + 1];

process[j + 1] = temp2;

}

}

}

int TAT[] = new int[numberOfProcess + 1];

int waitingTime[] = new int[numberOfProcess + 1];

// Calculating Waiting Time & Turn Around Time

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

TAT[i] = burstTime[i] + waitingTime[i];

waitingTime[i + 1] = TAT[i];

}

int totalWT = 0;

int totalTAT = 0;

double avgWT;

double avgTAT;

System.out.println("Process BT WT TAT");

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

System.out.println(process[i] + " " + burstTime[i] + " " + waitingTime[i] + " " + (TAT[i]));

totalTAT += (waitingTime[i] + burstTime[i]);

totalWT += waitingTime[i];

}

avgWT = totalWT / (double) numberOfProcess;

avgTAT = totalTAT / (double) numberOfProcess;

System.out.println("\n Average Wating Time: " + avgWT);

System.out.println(" Average Turn Around Time: " + avgTAT);

}

}

RR(preemptive)

1. **import** java.util.Scanner;
2. **public** **class** RoundRobin
3. {
4. **public** **static** **void** main(String args[])
5. {
6. **int** n,i,qt,count=0,temp,sq=0,bt[],wt[],tat[],rem\_bt[];
7. **float** awt=0,atat=0;
8. bt = **new** **int**[10];
9. wt = **new** **int**[10];
10. tat = **new** **int**[10];
11. rem\_bt = **new** **int**[10];
12. Scanner s=**new** Scanner(System.in);
13. System.out.print("Enter the number of process (maximum 10) = ");
14. n = s.nextInt();
15. System.out.print("Enter the burst time of the process\n");
16. **for** (i=0;i<n;i++)
17. {
18. System.out.print("P"+i+" = ");
19. bt[i] = s.nextInt();
20. rem\_bt[i] = bt[i];
21. }
22. System.out.print("Enter the quantum time: ");
23. qt = s.nextInt();
24. **while**(**true**)
25. {
26. **for** (i=0,count=0;i<n;i++)
27. {
28. temp = qt;
29. **if**(rem\_bt[i] == 0)
30. {
31. count++;
32. **continue**;
33. }
34. **if**(rem\_bt[i]>qt)
35. rem\_bt[i]= rem\_bt[i] - qt;
36. **else**
37. **if**(rem\_bt[i]>=0)
38. {
39. temp = rem\_bt[i];
40. rem\_bt[i] = 0;
41. }
42. sq = sq + temp;
43. tat[i] = sq;
44. }
45. **if**(n == count)
46. **break**;
47. }
48. System.out.print("--------------------------------------------------------------------------------");
49. System.out.print("\nProcess\t      Burst Time\t       Turnaround Time\t          Waiting Time\n");
50. System.out.print("--------------------------------------------------------------------------------");
51. **for**(i=0;i<n;i++)
52. {
53. wt[i]=tat[i]-bt[i];
54. awt=awt+wt[i];
55. atat=atat+tat[i];
56. System.out.print("\n "+(i+1)+"\t "+bt[i]+"\t\t "+tat[i]+"\t\t "+wt[i]+"\n");
57. }
58. awt=awt/n;
59. atat=atat/n;
60. System.out.println("\nAverage waiting Time = "+awt+"\n");
61. System.out.println("Average turnaround time = "+atat);
62. }
63. }

Assignment 2

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace maths3

{

public static class functions

{

public static int Add(int n1, int n2)

{

return n1 + n2;

}

public static int sub(int n1, int n2)

{

return n1 - n2;

}

public static int mul(int n1, int n2)

{

return n1 \* n2;

}

public static int div(int n1, int n2)

{

return n1 / n2;

}

}

}

Program code

(here in this project we use DLL file)

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using maths3;

namespace maths3App

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine(functions.Add(4, 10));

Console.WriteLine(functions.sub(4, 2));

Console.WriteLine(functions.mul(4, 10));

Console.WriteLine(functions.div(4, 2));

Console.ReadKey();

}

}

}