5. Write a program to compute the average waiting time and turnaround time based on Preemptive shortest remaining processing time first (SRPT) algorithm for the following set of processes, with the arrival times and the CPU-burst times given in milliseconds

Process Arrival Time Burst Time

P1 0 5

P2 1 3

P3 2 3

P4 4 1

Program:’

#include<stdio.h>

int main()

{

int burst\_time[20],p[20],waiting\_time[20],tat[20],i,j,n,total=0,pos,temp;

float avg\_waiting\_time,avg\_tat;

printf("please enter number of process: ");

scanf("%d",&n);

printf("\n enter the Burst Time:\n");

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

{

printf("p%d:",i+1);

scanf("%d",&burst\_time[i]);

p[i]=i+1;

}

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

{

pos=i;

for(j=i+1;j<n;j++)

{

if(burst\_time[j]<burst\_time[pos])

pos=j;

}

temp=burst\_time[i];

burst\_time[i]=burst\_time[pos];

burst\_time[pos]=temp;

temp=p[i];

p[i]=p[pos];

p[pos]=temp;

}

waiting\_time[0]=0;

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

{

waiting\_time[i]=0;

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

waiting\_time[i]+=burst\_time[j];

total+=waiting\_time[i];

}

avg\_waiting\_time=(float)total/n;

total=0;

printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");

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

{

tat[i]=burst\_time[i]+waiting\_time[i];

total+=tat[i];

printf("\np%d\t\t %d\t\t %d\t\t\t%d",p[i],burst\_time[i],waiting\_time[i],tat[i]);

}

avg\_tat=(float)total/n;

printf("\n\n the average Waiting Time=%f",avg\_waiting\_time);

printf("\n the average Turnaround Time=%f\n",avg\_tat);

}

Output:

enter number of process: 4

enter the Burst Time:

p1:5

p2:3

p3:3

p4:1

Process Burst Time Waiting Time Turnaround Time

p4 1 0 1

p2 3 1 4

p3 3 4 7

p1 5 7 12

the average Waiting Time=3.000000

the average Turnaround Time=6.000000

6. Write a C program to implement the deadlock detection algorithm for a system with 3 processes and 3 resource instances and the resource matrices are given below.

Max Matrix Allocation Matrix

3 6 8 3 3 3

4 3 3 2 0 3

3 4 4 1 2 4

The number of available resources is [1,2,0]. Determine if the system is in a deadlock state and identify the deadlocked processes.

Program:

#include<stdio.h>

static int mark[20];

int i, j, np, nr;

int main()

{

int alloc[10][10],request[10][10],avail[10],r[10],w[10];

printf ("\nEnter the no of the process: ");

scanf("%d",&np);

printf ("\nEnter the no of resources: ");

scanf("%d",&nr);

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

{

printf("\nTotal Amount of the Resource R % d: ",i+1);

scanf("%d", &r[i]);

}

printf("\nEnter the request matrix:");

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

for(j=0;j<nr;j++)

scanf("%d",&request[i][j]);

printf("\nEnter the allocation matrix:");

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

for(j=0;j<nr;j++)

scanf("%d",&alloc[i][j]);

for(j=0;j<nr;j++)

{

avail[j]=r[j];

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

{

avail[j]-=alloc[i][j];

}

}

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

{

int count=0;

for(j=0;j<nr;j++)

{

if(alloc[i][j]==0)

count++;

else

break;

}

if(count==nr)

mark[i]=1;

}

for(j=0;j<nr; j++)

w[j]=avail[j];

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

{

int canbeprocessed= 0;

if(mark[i]!=1)

{

for(j=0;j<nr;j++)

{

if(request[i][j]<=w[j])

canbeprocessed=1;

else

{

canbeprocessed=0;

break;

}

}

if(canbeprocessed)

{

mark[i]=1;

for(j=0;j<nr;j++)

w[j]+=alloc[i][j];

}

}

}

int deadlock=0;

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

if(mark[i]!=1)

deadlock=1;

if(deadlock)

printf("\n Deadlock detected");

else

printf("\n No Deadlock possible");

}

Output:

Enter the no of the process: 3

Enter the no of resources: 3

Total Amount of the Resource R 1: 1

Total Amount of the Resource R 2: 2

Total Amount of the Resource R 3: 0

Enter the request matrix:3

6

8

4

3

3

3

4

4

Enter the allocation matrix:3

3

3

2

0

3

1

2

4

Deadlock detected