NAME:D.SAI SATHWIK

ROLLNO:B66

SECTION:K17DW

REGNO:11716610

Ques. 12. Reena’s operating system uses an algorithm for deadlock avoidance to manage the allocation of resources say three namely A, B, and C to three processes P0, P1, and P2. Consider the following scenario as reference .user must enter the current state of system as given in this example :

Suppose P0 has 0,0,1 instances , P1 is having 3,2,0 instances and P2 occupies 2,1,1 instances of A,B,C resource respectively.

Also the maximum number of instances required for P0 is 8,4,3 and for p1 is 6,2,0 and finally for P2 there are 3,3,3 instances of resources A,B,C respectively. There are 3 instances of resource A, 2 instances of resource B and 2 instances of resource C available. Write a program to check whether Reena’s operating system is in a safe state or not in the following independent requests for additional resources in the

current state:

1. Request1: P0 requests 0 instances of A and 0 instances of B and 2 instances of C.

2. Request2: P1 requests for 2 instances of A, 0 instances of B and 0 instances of C.

All the request must be given by user as input.

#include<conio.h>

#include<stdio.h>

int main()

{

int n; //n number of process

int r; // number of resources

int i,j,k,cnt,cntt;

int avail[10],p[10];

int need[10][10],alloc[10][10],max[10][10];

printf("\nEnter number of process :");

scanf("%d",&n);

printf("\n Enter resources available : ");

scanf("%d",&r);

printf("\nEnter insatnces for resources :\n");

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

{ printf("R%d ",i+1);

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

}

printf("\n Enter allocation matrix \n");

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

{

printf("p%d",i+1); p[i]=0;

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

{

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

}

}

printf("\n Enter MAX matrix \n");

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

{

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

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

{

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

}

}

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

{

printf("\np%d\t",i+1) ;

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

{

need[i][j]=max[i][j]-alloc[i][j];

printf("\t%d",need[i][j]);

}

}

k=0; cntt=0;

printf("\n\n");

while(k<15)

{

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

{ cnt=0;

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

{

if(p[i]==1) break;

if(need[i][j]<=avail[j])

{

cnt++;

}

if(cnt==r)

{

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

{

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

}

printf("p%d\t",i+1); p[i]=1; cntt++;

}

}

} k++;

}

if(cntt<n-1)

{

printf("\n deadlock ");

}

getch();

}

Q39.Thereare five processes in the system. All five processes arrive at time 0, in the order

given, with the length of the CPU burst given in milliseconds:

Process burst time

P1 10

P2 29

P3 3

Pi 7

P5 12

Write a C program which will incorporate SFJ scheduling and print the ending time of

process Pi.

#include<stdio.h>

int main()

{

int i,n,p[10]={1,2,3,4,5,6,7,8,9,10},min,k=1,btime=0;

int bt[10],temp,j,at[10],wt[10],tt[10],ta=1,sum=0;

float wavg=0,tavg=0,tsum=0,wsum=0;

printf("Enter the No. of processes :\n");

scanf("%d",&n);

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

{

printf("Enter the Arrival time of p%d process :\n",i+1);

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

printf("Brust time of p%d process :\n",i+1);

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

}

/\*Sorting According to Arrival Time\*/

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

{

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

{

if(at[i]<at[j])

{

temp=p[j];

p[j]=p[i];

p[i]=temp;

temp=at[j];

at[j]=at[i];

at[i]=temp;

temp=bt[j];

bt[j]=bt[i];

bt[i]=temp;

}

}

}

/\*Arranging the table according to Burst time,

Execution time and Arrival Time

Arrival time <= Execution time

\*/

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

{

btime=btime+bt[j];

min=bt[k];

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

{

if (btime>=at[i] && bt[i]<min)

{

temp=p[k];

p[k]=p[i];

p[i]=temp;

temp=at[k];

at[k]=at[i];

at[i]=temp;

temp=bt[k];

bt[k]=bt[i];

bt[i]=temp;

}

}

k++;

}

wt[0]=0;

sum=1;

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

{

sum=sum+bt[i-1];

wt[i]=sum-at[i];

wsum=wsum+wt[i];

}

wavg=(wsum/n);

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

{

ta=ta+bt[i];

tt[i]=ta-at[i];

tsum=tsum+tt[i];

}

tavg=(tsum/n);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n RESULT:-");

printf("\nProcess\t Burst\t Arrival\t Waiting\t Turn-around-time" );

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

{

printf("\n p%d\t %d\t %d\t\t %d\t\t\t%d",p[i],bt[i],at[i],wt[i],tt[i]);

}

printf("\n\nAVERAGE WAITING TIME : %f",wavg);

printf("\nAVERAGE TURN AROUND TIME : %f",tavg);

return 0;

}