7. Design, develop and implement a C/C++/Java program to simulate the working of Shortest remaining time and Round Robin (RR) scheduling algorithms. Experiment with different quantum sizes for RR algorithm.

#include<stdio.h>

#include<stdlib.h>

int no;

void roundrobin(int,int,int[],int[]);

void srtf();

main()

{

int n,tq,choice;

int bt[10], st[10], i,j,k;

for(;;)

{

printf("Enter choice\n");

printf("1.Round Robin \n2.str \n3.Exit\n");

scanf("%d",&choice);

switch(choice)

{

case 1: printf("Round robin scheduling algorithm\n");

printf("Enter the number of process\n");

scanf("%d",&n);

printf("Enter burst time for sequences\n");

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

{

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

st[i]=bt[i];

}

printf("Enter time quantum\n");

scanf("%d",&tq);

roundrobin(n,tq,st,bt);

break;

case 2: printf("\n\n-------------------Shortest remaining time------");

srtf();

break;

case 3: exit(0);

break;

}

}

}

void roundrobin(int n, int tq, int st[], int bt[])

{

int time=0;

int tat[10],wt[10],i,count=0,swt=0,stat=0,temp1,sq=0,j,k;

float awt=0.0, atat=0.0;

while(1)

{

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

{

temp1=tq;

if(st[i]==0)

{

count++;

continue;

}

if (st[i]>tq)

st[i]=st[i]-tq;

else if(st[i]>=0)

{

temp1=st[i];

st[i]=0;

}

sq=sq+temp1;

tat[i]=sq;

}

if(n==count)

break;

}

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

{

wt[i]=tat[i]-bt[i];

swt=swt+wt[i];

stat=stat+tat[i];

}

awt=(float)swt/n;

atat=(float)stat/n;

printf("process no burst time waiting time turnaround time\n");

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

printf("%d \t\t %d \t\t %d \t\t %d \t\t\n",i+1,bt[i],wt[i],tat[i]);

printf("average waiting time is%f\n avg turnaround time is %f\n",awt,atat);

}

void srtf()

{

int n,j=0,st[10],bt[10],rt[10],remain=0,smallest,time=0,i,endtime,swt=0,stat=0;

printf("enter the no. of process:");

scanf("%d",&n);

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

{

printf("enter the arrivaltime for p[%d]:",i+1);

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

printf("enter the burst time for p[%d]:",i+1);

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

rt[i]=bt[i];

}

rt[100]=999;

printf("process\t|writing time\t|turnarround time\n");

for(time=0;remain!=n;time++)

{

smallest=100;

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

{

if(st[i]<=time && rt[i]<rt[smallest] && rt[i]>0)

{

smallest=i;

}

}

rt[smallest]--;

if(rt[smallest]==0)

{

remain++;

endtime=time+1;

j=smallest;

printf("p[%d]\t|\t%d\t|\t%d\n",smallest+1,endtime-bt[j]-st[j],endtime-st[j]);

swt +=endtime-bt[j]-st[j];

stat +=endtime-st[j];

}

}

float awt=0.0,atat=0.0;

awt=(float)swt/n;

atat=(float)stat/n;

printf("average waiting time:%f\n",awt);

printf("average turnarround time:%f\n",atat);

}

/\* **OUTPUT**

Enter choice

1.Round Robin

2.str

3.Exit

1

Round robin scheduling algorithm

Enter the number of process

4

Enter burst time for sequences

4 2 5 2

Enter time quantum

15

process no burst time waiting time turnaround time

1 4 0 4

2 2 4 6

3 5 6 11

4 2 11 13

average waiting time is5.250000

average turnaround time is 8.500000

Enter choice

1.Round Robin

2.str

3.Exit

2

-------------------Shortest remaining time----------------------

enter the no. of process:5

enter the arrivaltime for p[1]: 3

enter the burst time for p[1]: 5

enter the arrivaltime for p[2]: 0

enter the burst time for p[2]: 4

enter the arrivaltime for p[3]: 5

enter the burst time for p[3]: 4

enter the arrivaltime for p[4]: 7

enter the burst time for p[4]: 2

enter the arrivaltime for p[5]: 5

enter the burst time for p[5]: 4

process |writing time |turnarround time

p[2] | 0 | 4

p[1] | 1 | 6

p[4] | 2 | 4

p[3] | 6 | 10

p[5] | 10 | 14

average waiting time:3.800000

average turnarround time:7.600000

Enter choice

1.Round Robin

2.str

3.Exit

\*/

8. Design, develop and implement a C/C++/Java program to implement Banker’s algorithm. Assume suitable input required to demonstrate the results.

#include<stdio.h>

struct process

{

int all[6],max[6],need[6],finished,request[6];

}p[10];

int avail[6],sseq[10],ss=0,check1=0,check2=0,n,pid,nor,nori,work[6];

int main()

{

int safeseq(void);

int ch,k,i=0,j=0,pid,ch1;

int violationcheck=0,waitcheck=0;

do

{

printf("\n1.Input\n2.New Request\n3.Safe State or Not\n4.Print\n5.Exit\nEnter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 1: printf("\nEnter the number of processes:");

scanf("%d",&n);

printf("\nEnter the number of resources:");

scanf("%d",&nor);

printf("\nEnter the available resources:");

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

{

for(k=0;k<nor;k++)

{

if(j==0)

{

printf("\nFor Resource Type %d:",k);

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

}

p[j].max[k]=0;

p[j].all[k]=0;

p[j].need[k]=0;

p[j].finished=0;

p[j].request[k]=0;

}

}

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

{

printf("\nEnter Max and Allocated Resources for P %d :",i);

for(j=0;j<nor;j++)

{

printf("\nEnter the Max of Resources %d:",j);

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

printf("\nAllocation of Resources %d:",j);

scanf("%d",&p[i].all[j]);

if(p[i].all[j]>p[i].max[j])

{

printf("\nAllocation should be less than or equal to Max\n");

j--;

}

else

p[i].need[j]=p[i].max[j]-p[i].all[j];

}

}

break;

case 2:

violationcheck=0;

waitcheck=0;

printf("\nRequesting Process ID:\n");

scanf("%d",&pid);

for(j=0;j<nor;j++)

{

printf("\nNumber of Request for Resource %d:",j);

scanf("%d",&p[pid].request[j]);

if(p[pid].request[j]>p[pid].need[j])

violationcheck=1;

if(p[pid].request[j]>avail[j])

waitcheck=1;

}

if(violationcheck==1)

printf("\nThe Process Exceeds its Max needs: Terminated\n");

else if(waitcheck==1)

printf("\nLack of Resources: Process State - Wait\n");

else

{

for(j=0;j<nor;j++)

{

avail[j]=avail[j]-p[pid].request[j];

p[pid].all[j]=p[pid].all[j]+p[pid].request[j];

p[pid].need[j]=p[pid].need[j]-p[pid].request[j];

}

ch1=safeseq();

if(ch1==0)

{

for(j=0;j<nor;j++)

{

avail[j]=avail[j]+p[pid].request[j];

p[pid].all[j]=p[pid].all[j]-p[pid].request[j];

p[pid].need[j]=p[pid].need[j]+p[pid].request[j];

}

}

else if(ch1==1)

printf("\nRequest committed.\n");

}

break;

case 3:

if(safeseq()==1)

printf("\nThe System is in Safe State\n");

else

printf("\nThe System is not in Safe State\n");

break;

case 4:

printf("\nNumber of Process:%d\n",n);

printf("\nNumber of Resources:%d\n",nor);

printf("\nPid\tMax\tAllocated\tNeed\n");

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

{

printf(" P%d :",i);

for(j=0;j<nor;j++)

printf(" %d ",p[i].max[j]);

printf("\t");

for(j=0;j<nor;j++)

printf(" %d ",p[i].all[j]);

printf("\t");

for(j=0;j<nor;j++)

printf(" %d ",p[i].need[j]);

printf("\n");

}

printf("\nAvailable:\n");

for(i=0;i<nor;i++)

printf(" %d ",avail[i]);

break;

case 5: break;

}

}while(ch!=5);

return 0;

}

int safeseq()

{

int tj,tk,i,j,k;

ss=0;

for(j=0;j<nor;j++)

work[j]=avail[j];

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

p[j].finished=0;

for(tk=0;tk<nor;tk++)

{

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

{

if(p[j].finished==0)

{

check1=0;

for(k=0;k<nor;k++)

if(p[j].need[k]<=work[k])

check1++;

if(check1==nor)

{

for(k=0;k<nor;k++)

{

work[k]=work[k]+p[j].all[k];

p[j].finished=1;

}

sseq[ss]=j;

ss++;

}

}

}

}

check2=0;

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

if(p[i].finished==1)

check2++;

printf("\n");

if(check2>=n)

{

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

printf("p%d",sseq[tj]);

return 1;

}

else

printf("\nThe System is not in Safe State\n");

return 0;

}

/\* **OUTPUT**

1.Input

2.New Request

3.Safe State or Not

4.Print

5.Exit

Enter your choice:1

Enter the number of processes:5

Enter the number of resources:3

Enter the available resources:

For Resource Type 0:3

For Resource Type 1:3

For Resource Type 2:2

Enter Max and Allocated Resources for P 0 :

Enter the Max of Resources 0:7

Allocation of Resources 0:0

Enter the Max of Resources 1:5

Allocation of Resources 1:1

Enter the Max of Resources 2:3

Allocation of Resources 2:0

Enter Max and Allocated Resources for P 1 :

Enter the Max of Resources 0:3

Allocation of Resources 0:2

Enter the Max of Resources 1:2

Allocation of Resources 1:0

Enter the Max of Resources 2:2

Allocation of Resources 2:0

Enter Max and Allocated Resources for P 2 :

Enter the Max of Resources 0:9

Allocation of Resources 0:3

Enter the Max of Resources 1:0

Allocation of Resources 1:0

Enter the Max of Resources 2:2

Allocation of Resources 2:2

Enter Max and Allocated Resources for P 3 :

Enter the Max of Resources 0:2

Allocation of Resources 0:2

Enter the Max of Resources 1:2

Allocation of Resources 1:1

Enter the Max of Resources 2:2

Allocation of Resources 2:1

Enter Max and Allocated Resources for P 4 :

Enter the Max of Resources 0:4

Allocation of Resources 0:0

Enter the Max of Resources 1:3

Allocation of Resources 1:0

Enter the Max of Resources 2:3

Allocation of Resources 2:2

1.Input

2.New Request

3.Safe State or Not

4.Print

5.Exit

Enter your choice:3

p1p3p4p0p2

The System is in Safe State

1.Input

2.New Request

3.Safe State or Not

4.Print

5.Exit

Enter your choice:4

Number of Process:5

Number of Resources:3

Pid Max Allocated Need

P0 : 7 5 3 0 1 0 7 4 3

P1 : 3 2 2 2 0 0 1 2 2

P2 : 9 0 2 3 0 2 6 0 0

P3 : 2 2 2 2 1 1 0 1 1

P4 : 4 3 3 0 0 2 4 3 1

Available:

3 3 2

1.Input

2.New Request

3.Safe State or Not

4.Print

5.Exit

Enter your choice:2

Requesting Process ID:

1

Number of Request for Resource 0:1

Number of Request for Resource 1:0

Number of Request for Resource 2:2

p1p3p4p0p2

Request committed.

1.Input

2.New Request

3.Safe State or Not

4.Print

5.Exit

Enter your choice:4

Number of Process:5

Number of Resources:3

Pid Max Allocated Need

P0 : 7 5 3 0 1 0 7 4 3

P1 : 3 2 2 3 0 2 0 2 0

P2 : 9 0 2 3 0 2 6 0 0

P3 : 2 2 2 2 1 1 0 1 1

P4 : 4 3 3 0 0 2 4 3 1

Available:

2 3 0

1.Input

2.New Request

3.Safe State or Not

4.Print

5.Exit

Enter your choice:3

p1p3p4p0p2

The System is in Safe State

1.Input

2.New Request

3.Safe State or Not

4.Print

5.Exit

Enter your choice:2

Requesting Process ID:

4

Number of Request for Resource 0:3

Number of Request for Resource 1:3

Number of Request for Resource 2:0

Lack of Resources: Process State - Wait

1.Input

2.New Request

3.Safe State or Not

4.Print

5.Exit

Enter your choice:2

Requesting Process ID:

0

Number of Request for Resource 0:0

Number of Request for Resource 1:2

Number of Request for Resource 2:0

The System is not in Safe State

1.Input

2.New Request

3.Safe State or Not

4.Print

5.Exit

Enter your choice:4

Number of Process:5

Number of Resources:3

Pid Max Allocated Need

P0 : 7 5 3 0 1 0 7 0 3

P1 : 3 2 2 3 0 2 0 2 0

P2 : 9 0 2 3 0 2 6 0 0

P3 : 2 2 2 2 1 1 0 1 1

P4 : 4 3 3 0 0 2 4 3 1

Available:

2 3 0

1.Input

2.New Request

3.Safe State or Not

4.Print

5.Exit

Enter your choice:5

\*/

9. Design, develop and implement a C/C++/Java program to implement page replacement algorithms LRU and FIFO. Assume suitable input required to demonstrate the results.

#include<stdio.h>

#include<stdlib.h>

void FIFO(char [ ],char [ ],int,int);

void lru(char [ ],char [ ],int,int);

void opt(char [ ],char [ ],int,int);

int main()

{

int ch,YN=1,i,l,f;

char F[10],s[25];

printf("\n\n\tEnter the no of empty frames: ");

scanf("%d",&f);

printf("\n\n\tEnter the length of the string: ");

scanf("%d",&l);

printf("\n\n\tEnter the string: ");

scanf("%s",s);

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

F[i]=-1;

do

{

printf("\n\n\t\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*\*\*");

printf("\n\n\t1:FIFO\n\n\t2:LRU \n\n\t4:EXIT");

printf("\n\n\tEnter your choice: ");

scanf("%d",&ch);

switch(ch)

{

case 1:

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

{

F[i]=-1;

}

FIFO(s,F,l,f);

break;

case 2:

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

{

F[i]=-1;

}

lru(s,F,l,f);

break;

case 4:

exit(0);

}

printf("\n\n\tDo u want to continue IF YES PRESS 1\n\n\tIF NO PRESS 0 : ");

scanf("%d",&YN);

}

while(YN==1);return(0);

}

//FIFO

void FIFO(char s[],char F[],int l,int f)

{

int i,j=0,k,flag=0,cnt=0;

printf("\n\tPAGE\t FRAMES\t FAULTS");

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

{

for(k=0;k<f;k++)

{

I f(F[k]==s[i])

flag=1;

}

if(flag==0)

{

printf("\n\t%c\t",s[i]);

F[j]=s[i];

j++;

for(k=0;k<f;k++)

{

printf(" %c",F[k]);

}

printf("\tPage-fault%d",cnt);

cnt++;

}

else

{

flag=0;

printf("\n\t%c\t",s[i]);

for(k=0;k<f;k++)

{

printf(" %c",F[k]);

}

printf("\tNo page-fault");

}

if(j==f)

j=0;

}

}

//LRU

void lru(char s[],char F[],int l,int f)

{

int i,j=0,k,m,flag=0,cnt=0,top=0;

printf("\n\tPAGE\t FRAMES\t FAULTS");

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

{

for(k=0;k<f;k++)

{

if(F[k]==s[i])

{

flag=1;

break;

}

}

printf("\n\t%c\t",s[i]);

if(j!=f && flag!=1)

{

F[top]=s[i];

j++;

if(j!=f)

top++;

}

else

{

if(flag!=1)

{

for(k=0;k<top;k++)

{

F[k]=F[k+1];

}

F[top]=s[i];

}

if(flag==1)

{

for(m=k;m<top;m++)

{

F[m]=F[m+1];

}

F[top]=s[i];

}

}

for(k=0;k<f;k++)

{

printf(" %c",F[k]);

}

if(flag==0)

{

printf("\tPage-fault%d",cnt);

cnt++;

}

else

printf("\tNo page fault");

flag=0;

}

}

**OUTPUT**

Enter the no of empty frames: 3

Enter the length of the string: 20

Enter the string: 70120304230321201701

\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*\*\*

1:FIFO

2:LRU

4:EXIT

Enter your choice: 1

PAGE FRAMES FAULTS

7 7 Page-fault0

0 7 0 Page-fault1

1 7 0 1 Page-fault2

2 2 0 1 Page-fault3

0 2 0 1 No page-fault

3 2 3 1 Page-fault4

0 2 3 0 Page-fault5

4 4 3 0 Page-fault6

2 4 2 0 Page-fault7

3 4 2 3 Page-fault8

0 0 2 3 Page-fault9

3 0 2 3 No page-fault

2 0 2 3 No page-fault

1 0 1 3 Page-fault10

2 0 1 2 Page-fault11

0 0 1 2 No page-fault

1 0 1 2 No page-fault

7 7 1 2 Page-fault12

0 7 0 2 Page-fault13

1 7 0 1 Page-fault14

Do u want to continue IF YES PRESS 1

IF NO PRESS 0 : 1

\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*\*\*

1:FIFO

2:LRU

4:EXIT

Enter your choice: 2

PAGE FRAMES FAULTS

7 7 Page-fault0

0 7 0 Page-fault1

1 7 0 1 Page-fault2

2 0 1 2 Page-fault3

0 1 2 0 No page fault

3 2 0 3 Page-fault4

0 2 3 0 No page fault

4 3 0 4 Page-fault5

2 0 4 2 Page-fault6

3 4 2 3 Page-fault7

0 2 3 0 Page-fault8

3 2 0 3 No page fault

2 0 3 2 No page fault

1 3 2 1 Page-fault9

2 3 1 2 No page fault

0 1 2 0 Page-fault10

1 2 0 1 No page fault

7 0 1 7 Page-fault11

0 1 7 0 No page fault

1 7 0 1 No page fault

Do u want to continue IF YES PRESS 1

IF NO PRESS 0 :