#include<stdio.h>

void readInput();

void FIFO();

void OPTIMAL();

void LRU();

void LFU();

int inarray(char mem[],char c);

int n,nf,nff;

int max(int temp[]);

char refstr[100];

void settemp(int temp[],int val);

void setvisit(int visit[]);

int detect(int visit[],int val,int temp[],char mem[]);

int min(int temp[]);

void main()

{

int option=0;

while(option!=6)

{

printf("ENTER option\n1.READ\_INPUT\n2.FIFO\n3.OPTIMAL\n4.LRU\n5.LFU\n6.EXIT\nCHOOSE OPTION : ");

scanf("%d",&option);

if(option==1)

{

readInput();

}

else if(option==2)

{

FIFO();

}

else if(option==3)

{

OPTIMAL();

}

else if(option==4)

{

LRU();

}

else if(option==5)

{

LFU();

}

else if(option!=6)

{

printf("Choose a valid option\n");

}

}

}

void readInput()

{

printf("Enter the number of free frames : ");

scanf("%d",&nff);

printf("Enter the no of frames required by the process : ");

scanf("%d",&nf);

printf("Enter the length of the refence string : ");

scanf("%d",&n);

int i;

printf("Enter the Reference string : ");

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

{

scanf(" %c",&refstr[i]);

}

}

void FIFO()

{

int pf=0,k=0,i,j;

char mem[nf];

printf("\t\t FIFO Page Replacement Algorithm\n\nThe Reference string : ");

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

{

printf("%c ",refstr[i]);

}

printf("\nPage ref-> MEMORY ->PF\n");

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

{

if(i<nf)

{

pf++;

printf("%c\t -> ",refstr[i]);

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

{

printf("%c ",refstr[j]);

}

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

{

printf("- ");

}

printf("\t-> %d\n",pf);

mem[i]=refstr[i];

}

else

{

if(inarray(mem,refstr[i]))

{

printf("%c\t -> ",refstr[i]);

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

{

printf("%c ",mem[j]);

}

printf("\t-> -\n");

}

else

{

pf++;

mem[k]=refstr[i];

k++;

if(k==nf)

{

k=0;

}

printf("%c\t -> ",refstr[i]);

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

{

printf("%c ",mem[j]);

}

printf("\t-> %d\n",pf);

}

}

}

printf("The number of page faults is %d\n",pf);

}

int inarray(char mem[],char c)

{

int i;

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

{

if(c==mem[i])

{

return i+1;

}

}

return 0;

}

void disp(int temp[],int a)

{

int i;

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

{

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

}

printf("\n");

}

void OPTIMAL()

{

int pf=0,k=0,i,j,m,temp[nf],r=1;

char mem[nf];

int visit[n];

setvisit(visit);

printf("\t\t OPTIMAL Page Replacement Algorithm\n\nThe Reference string : ");

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

{

printf("%c ",refstr[i]);

}

printf("\nPage ref -> MEMORY ->PF\n");

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

{

if(i<nf)

{

pf++;

printf("%c\t -> ",refstr[i]);

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

{

printf("%c ",refstr[j]);

}

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

{

printf("- ");

}

printf("\t-> %d\n",pf);

mem[i]=refstr[i];

}

else

{

if(inarray(mem,refstr[i]))

{

printf("%c\t -> ",refstr[i]);

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

{

printf("%c ",mem[j]);

}

printf("\t-> -\n");

}

else

{

pf++;

settemp(temp,1000);

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

{

m=inarray(mem,refstr[j]);

if(m!=0)

{

if(temp[m-1]==1000)

{

temp[m-1]=r;

r++;

}

}

}

r=1;

k=detect(visit,max(temp),temp,mem);

mem[k]=refstr[i];

printf("%c\t -> ",refstr[i]);

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

{

printf("%c ",mem[j]);

}

printf("\t-> %d\n",pf);

}

}

}

printf("The number of page faults is %d\n",pf);

}

int detect(int visit[],int val,int temp[],char mem[])

{

int i,j;

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

{

if(visit[i]==0)

{

j=inarray(mem,refstr[i]);

if(j!=0 && temp[j-1]==val)

{

visit[i]=1;

return j-1;

}

}

}

}

void settemp(int temp[],int val)

{

int i;

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

{

temp[i]=val;

}

}

void setvisit(int visit[])

{

int i;

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

{

visit[i]=0;

}

}

int max(int temp[])

{

int i,large=temp[0];

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

{

if(temp[i]>large)

{

large=temp[i];

}

}

return large;

}

void LRU()

{

int pf=0,k=0,i,j,m,temp[nf],r=1;

char mem[nf];

int visit[n];

setvisit(visit);

printf("\t\t OPTIMAL Page Replacement Algorithm\n\nThe Reference string : ");

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

{

printf("%c ",refstr[i]);

}

printf("\nPage ref -> MEMORY ->PF\n");

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

{

if(i<nf)

{

pf++;

printf("%c\t -> ",refstr[i]);

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

{

printf("%c ",refstr[j]);

}

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

{

printf("- ");

}

printf("\t-> %d\n",pf);

mem[i]=refstr[i];

}

else

{

if(inarray(mem,refstr[i]))

{

printf("%c\t -> ",refstr[i]);

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

{

printf("%c ",mem[j]);

}

printf("\t-> -\n");

}

else

{

pf++;

settemp(temp,1000);

for(j=i-1;j>=0;j--)

{

m=inarray(mem,refstr[j]);

if(m!=0)

{

if(temp[m-1]==1000)

{

temp[m-1]=r;

r++;

}

}

}

r=1;

k=detect(visit,max(temp),temp,mem);

mem[k]=refstr[i];

printf("%c\t -> ",refstr[i]);

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

{

printf("%c ",mem[j]);

}

printf("\t-> %d\n",pf);

}

}

}

printf("The number of page faults is %d\n",pf);

}

void LFU()

{

int pf=0,k=0,i,j,m,temp[nf],r=1;

char mem[nf];

int visit[n];

setvisit(visit);

printf("\t\t OPTIMAL Page Replacement Algorithm\n\nThe Reference string : ");

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

{

printf("%c ",refstr[i]);

}

printf("\nPage ref -> MEMORY ->PF\n");

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

{

if(i<nf)

{

pf++;

printf("%c\t -> ",refstr[i]);

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

{

printf("%c ",refstr[j]);

}

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

{

printf("- ");

}

printf("\t-> %d\n",pf);

mem[i]=refstr[i];

}

else

{

if(inarray(mem,refstr[i]))

{

printf("%c\t -> ",refstr[i]);

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

{

printf("%c ",mem[j]);

}

printf("\t-> -\n");

}

else

{

pf++;

settemp(temp,0);

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

{

m=inarray(mem,refstr[j]);

if(m!=0)

{

temp[m-1]+=1;

}

}

r=1;

k=detect(visit,min(temp),temp,mem);

mem[k]=refstr[i];

printf("%c\t -> ",refstr[i]);

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

{

printf("%c ",mem[j]);

}

printf("\t-> %d\n",pf);

}

}

}

printf("The number of page faults is %d\n",pf);

}

int min(int temp[])

{

int i,small=temp[0];

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

{

if(temp[i]<small)

{

small=temp[i];

}

}

return small;

}

/\*

OUTPUT

PS F:\SEM4\OS\Assignment10> ./k

ENTER option

1.READ\_INPUT

2.FIFO

3.OPTIMAL

4.LRU

5.LFU

6.EXIT

CHOOSE OPTION : 1

Enter the number of free frames : 10

Enter the no of frames required by the process : 4

Enter the length of the refence string : 20

Enter the Reference string : 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

ENTER option

1.READ\_INPUT

2.FIFO

3.OPTIMAL

4.LRU

5.LFU

6.EXIT

CHOOSE OPTION : 2

FIFO Page Replacement Algorithm

The Reference string : 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

Page ref-> MEMORY ->PF

7 -> 7 - - - -> 1

0 -> 7 0 - - -> 2

1 -> 7 0 1 - -> 3

2 -> 7 0 1 2 -> 4

0 -> 7 0 1 2 -> -

3 -> 3 0 1 2 -> 5

0 -> 3 0 1 2 -> -

4 -> 3 4 1 2 -> 6

2 -> 3 4 1 2 -> -

3 -> 3 4 1 2 -> -

0 -> 3 4 0 2 -> 7

3 -> 3 4 0 2 -> -

2 -> 3 4 0 2 -> -

1 -> 3 4 0 1 -> 8

2 -> 2 4 0 1 -> 9

0 -> 2 4 0 1 -> -

1 -> 2 4 0 1 -> -

7 -> 2 7 0 1 -> 10

0 -> 2 7 0 1 -> -

1 -> 2 7 0 1 -> -

The number of page faults is 10

ENTER option

1.READ\_INPUT

2.FIFO

3.OPTIMAL

4.LRU

5.LFU

6.EXIT

CHOOSE OPTION : 3

OPTIMAL Page Replacement Algorithm

The Reference string : 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

Page ref -> MEMORY ->PF

7 -> 7 - - - -> 1

0 -> 7 0 - - -> 2

1 -> 7 0 1 - -> 3

2 -> 7 0 1 2 -> 4

0 -> 7 0 1 2 -> -

3 -> 3 0 1 2 -> 5

0 -> 3 0 1 2 -> -

4 -> 3 0 4 2 -> 6

2 -> 3 0 4 2 -> -

3 -> 3 0 4 2 -> -

0 -> 3 0 4 2 -> -

3 -> 3 0 4 2 -> -

2 -> 3 0 4 2 -> -

1 -> 1 0 4 2 -> 7

2 -> 1 0 4 2 -> -

0 -> 1 0 4 2 -> -

1 -> 1 0 4 2 -> -

7 -> 1 0 4 7 -> 8

0 -> 1 0 4 7 -> -

1 -> 1 0 4 7 -> -

The number of page faults is 8

ENTER option

1.READ\_INPUT

2.FIFO

3.OPTIMAL

4.LRU

5.LFU

6.EXIT

CHOOSE OPTION : 4

OPTIMAL Page Replacement Algorithm

The Reference string : 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

Page ref -> MEMORY ->PF

7 -> 7 - - - -> 1

0 -> 7 0 - - -> 2

1 -> 7 0 1 - -> 3

2 -> 7 0 1 2 -> 4

0 -> 7 0 1 2 -> -

3 -> 3 0 1 2 -> 5

0 -> 3 0 1 2 -> -

4 -> 3 0 4 2 -> 6

2 -> 3 0 4 2 -> -

3 -> 3 0 4 2 -> -

0 -> 3 0 4 2 -> -

3 -> 3 0 4 2 -> -

2 -> 3 0 4 2 -> -

1 -> 3 0 1 2 -> 7

2 -> 3 0 1 2 -> -

0 -> 3 0 1 2 -> -

1 -> 3 0 1 2 -> -

7 -> 7 0 1 2 -> 8

0 -> 7 0 1 2 -> -

1 -> 7 0 1 2 -> -

The number of page faults is 8

ENTER option

1.READ\_INPUT

2.FIFO

3.OPTIMAL

4.LRU

5.LFU

6.EXIT

CHOOSE OPTION : 5

OPTIMAL Page Replacement Algorithm

The Reference string : 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

Page ref -> MEMORY ->PF

7 -> 7 - - - -> 1

0 -> 7 0 - - -> 2

1 -> 7 0 1 - -> 3

2 -> 7 0 1 2 -> 4

0 -> 7 0 1 2 -> -

3 -> 3 0 1 2 -> 5

0 -> 3 0 1 2 -> -

4 -> 3 0 4 2 -> 6

2 -> 3 0 4 2 -> -

3 -> 3 0 4 2 -> -

0 -> 3 0 4 2 -> -

3 -> 3 0 4 2 -> -

2 -> 3 0 4 2 -> -

1 -> 3 0 1 2 -> 7

2 -> 3 0 1 2 -> -

0 -> 3 0 1 2 -> -

1 -> 3 0 1 2 -> -

7 -> 7 0 1 2 -> 8

0 -> 7 0 1 2 -> -

1 -> 7 0 1 2 -> -

The number of page faults is 8

ENTER option

1.READ\_INPUT

2.FIFO

3.OPTIMAL

4.LRU

5.LFU

6.EXIT

CHOOSE OPTION : 6

\*/