/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

OS Task

3rd Computer Engineering,Helwan University

1- ايه معوض راشد (030)

2- ريهام محمد ابو اليزيد (043)

3- حسن ابراهيم فتوح (038)

4- مصطفى مجدى احمد عثمان (097)

5- هبة اشرف فؤاد طه (105)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream> /\* Input & Output \*/

#include <stdlib.h> /\* Standard Library \*/

using namespace std;

int getReplaceposition(int counter[],int n)

{

int max = counter[0];

int pos=0;

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

{

if(counter[i]>max)

{

pos=i;

max = counter[i];

}

}

return pos;

}

// FIFO function

void FIFO(int pages[], int nPages, int nFrames)

{

// Complete this function

int flag, hlt,Totalhlt=0;

int pageFault=0;

// int \*pages = new int[nPages];

int \*frames = new int[nFrames];

int \*counter = new int[nFrames];

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

{

frames[i] = 0;

counter[i] = 0; //here 0 referes an empty space in frame

}

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

{

flag =0;

hlt = 0 ;

for(int j=0; j<nFrames; j++)

{

if(frames[j] == pages[i])

{

flag=1; //if page is present in frame (flag=1)

hlt = 1;

Totalhlt++;

break;

}

}

//if page is not present in frame (flag=0)

if(flag == 0)

{

pageFault++;

for(int j=0; j<nFrames; j++)

{

if(frames[j] == 0)

{

frames[j] = pages[i];

flag=1;

hlt = 0 ;

counter[j]++;

break;

}

}

}

//if there is no empty frame

if(flag == 0)

{

int pos = getReplaceposition(counter,nFrames);

frames[pos] = pages[i];

counter[pos] = 1;

for(int k=0; k<nFrames; k++)

{

if(k!=pos)

counter[k]++;

}

}

cout<<endl;

for(int j=0; j<nFrames; j++)

{

if (hlt == 1)

{

cout << " ";

}

else

{

cout<<frames[j]<<" ";

}

}

}

cout<<"\nTotal Hlt: "<< Totalhlt;

cout<<"\nTotal Miss: "<<pageFault;

}

// LFU function

void LFU(int arr[], int nPages, int nFrames)

{

int p;

bool done;

int totalMiss = 0;

int \*frames = new int [nFrames] ; /\* array for frames \*/

int \*frequency =new int [nFrames]; /\* array to check frequency for each page \*/

int \*check = new int [nPages]; /\* array to be checked if page leave memory or not \*/

int totalHlt = 0;

//initialize frames as empty

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

{

frames[i]= -1;

}

// initialize all frequency with 0 for expected pages 1-10

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

{

frequency[i]=0;

}

// initialize check bit for each page

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

{

check[i]= -1;

}

for (int readyPage=0; readyPage<nPages; readyPage++)

{

done = false; // to check if page finds a frame

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

{

// check if page is already exist

if (arr[readyPage]==frames[i])

{

totalHlt ++;

// increase frequency of the page

frequency[i]++;

done = true;

break;

}

// you find empty frame

else if (frames[i]== -1)

{

totalMiss++;

frames[i] = arr[readyPage];

frequency[i]++;

done = true;

break;

}

}

// you have to swap with another page

if (done==false)

{

int least= frequency[0]; /\* least as value \*/

int leastFrequentlyUsed = 0;/\* least as frame index \*/

// find frequency of current pages in the memory

for (int k=0 ; k<nFrames; k++)

{

// you find the least

if(frequency[k]<least)

{

least = frequency[k];

leastFrequentlyUsed = k ;

p = k;

}

// you find more than one page has the same frequency

else if (frequency[k]==least )

{

// check if the page leave the memory before

for (int j = 0; j<readyPage; j++)

{

// find first in

if(arr[j] == frames[leastFrequentlyUsed] && check[j]!=0)

{

p = j; // save swapped page

break;

}

else if (arr[j]==frames[k] && check[j]!=0)

{

least = frequency[k];

leastFrequentlyUsed = k ;

p = j; // save swapped page

break;

}

}

}

}

// swap with the least or first in

frames[leastFrequentlyUsed] = arr[readyPage];

done = true;

frequency[leastFrequentlyUsed] = 1;

check[p] = 0; //page leaved memory

totalMiss++;

}

for (int qq = 0 ; qq<nFrames; qq++)

cout<< frames[qq]<<" ";

cout<< "\n";

// end of if statment

}// end of for loop

cout << "total miss: "<<totalMiss<< "\n";

cout << "total HLT: "<<totalHlt<< "\n";

} // end of function

// minimum Freq.

int min(int counter[],int nFrames)

{

int minimum = counter[0];

int pos = 0;

for(int i=1;i<nFrames;i++) { if(minimum > counter[i])

{ minimum = counter[i];

pos = i;

}

}

return pos;

}

// LRU function

void LRU(int arr[], int ArraySize, int NFrames){

// Complete this function

int Frames[NFrames]; /\* The Array Of Frames That We have \*/

int counter[ArraySize],recent = 0;

int pageFault = 0;

int PageHLT =0;

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

{

Frames[i] = 0;

counter[i] = 0;//here 0 referes an empty space in frame

}

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

{ int flag =0, HLTflag=0;

for(int j=0;j<NFrames;j++)

{

if(Frames[j] == arr[i])

{flag=1;

counter[j] = recent++; //counter holds which frame is recently used,

//recently used page in frame will have a bigger number

//and least recently used page in frame will have a lower number

HLTflag =1 ;

break;

}

}

if(flag == 0)

{

for(int j=0;j<NFrames;j++)

{if(Frames[j] == 0)

{ Frames[j] = arr[i];

counter[j] = recent++;

flag=1;

pageFault++;

break;

}

}

}

if(flag == 0){

int PositionToreplace = min(counter,NFrames);

Frames[PositionToreplace] = arr[i];

counter[PositionToreplace] = recent++;

pageFault++;

}

//print frames

cout<<endl;

for(int j=0;j<NFrames;j++)

{

if(HLTflag == 1){

PageHLT++;

break;

}

cout<<Frames[j]<<" ";

}

}

cout<<"\nNumber Of Page HLT: "<<PageHLT;

cout<<"\nTotal Miss: "<<pageFault;

}

void MFU(int arr[], int nPages, int nFrames)

{

int p;

bool done;

int totalMiss = 0;

int \*frames = new int [nFrames] ; /\* array for frames \*/

int \*frequency =new int [nFrames]; /\* array to check frequency for each page \*/

int \*check = new int [nPages]; /\* array to be checked if page leave memory or not \*/

int totalHlt = 0;

//initialize frames as empty

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

{

frames[i]= -1;

}

// initialize all frequency with 0 for expected pages 1-10

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

{

frequency[i]=0;

}

// initialize check bit for each page

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

{

check[i]= -1;

}

for (int readyPage=0; readyPage<nPages; readyPage++)

{

done = false; // to check if page finds a frame

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

{

// check if page is already exist

if (arr[readyPage]==frames[i])

{

totalHlt ++;

// increase frequency of the page

frequency[i]++;

done = true;

break;

}

// you find empty frame

else if (frames[i]== -1)

{

totalMiss++;

frames[i] = arr[readyPage];

frequency[i]++;

done = true;

break;

}

}

// you have to swap with another page

if (done==false)

{

int Most= frequency[0]; /\* Most as value \*/

int MostFrequentlyUsed = 0;/\* Most as frame index \*/

// find frequency of current pages in the memory

for (int k=0 ; k<nFrames; k++)

{

// you find the Most

if(frequency[k]>Most)

{

Most = frequency[k];

MostFrequentlyUsed = k ;

p = k;

}

// you find more than one page has the same frequency

else if (frequency[k]==Most )

{

// check if the page leave the memory before

for (int j = 0; j<readyPage; j++)

{

// find first in

if(arr[j] == frames[MostFrequentlyUsed] && check[j]!=0)

{

p = j; // save swapped page

break;

}

else if (arr[j]==frames[k] && check[j]!=0)

{

Most = frequency[k];

MostFrequentlyUsed = k ;

p = j; // save swapped page

break;

}

}

}

}

// swap with the Most or first in

frames[MostFrequentlyUsed] = arr[readyPage];

done = true;

frequency[MostFrequentlyUsed] = 1;

check[p] = 0; //page leaved memory

totalMiss++;

}

for (int qq = 0 ; qq<nFrames; qq++)

cout<< frames[qq]<<" ";

cout<< "\n";

// end of if statment

}// end of for loop

cout << "total miss: "<<totalMiss<< "\n";

cout << "total HLT: "<<totalHlt<< "\n";

} // end of function

// Optimal function

void Optimal(int Pages[], int NPages, int NFrames){

// Frames Array

int \*Frames;

Frames = new int [NFrames];

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

Frames[i] = -1; // Empty Frame

int TotalMiss = 0; // Total Miss Counter

// Loop on Pages

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

{

bool isThereEmptyFrame = false;

bool isPageAlreadyPresented = false;

// Loop on Frames

for(int j=0; j<NFrames; j++)

{

// Check if the Page is aleardy presented

if(Frames[j] == Pages[i])

{

isPageAlreadyPresented = true;

break;

}

// Check if there is Empty Frame

else if(Frames[j] == -1)

{

TotalMiss++;

Frames[j] = Pages[i];

isThereEmptyFrame = true;

break;

}

} // End of Loop on Frames

// Need to Replace

if((!isThereEmptyFrame) && (!isPageAlreadyPresented))

{

TotalMiss++;

int MaxDistance = 0;

int Index = -1;

// Loop on Frames

for(int j=0; j< NFrames; j++)

{

bool isPageUsedInFuture = false;

// Loop on Future use Pages

for(int k=i+1; k<NPages; k++)

{

// is Page Used In Future

if(Frames[j] == Pages[k])

{

isPageUsedInFuture = true;

if((k - i) > MaxDistance)

{

MaxDistance = k - i;

Index = j;

}

break;

}

} // End Loop on Future use Pages

if(!isPageUsedInFuture)

{

MaxDistance = NPages; // The Biggest Value forever

Index = j;

break;

}

} // End of Loop on Frames

// Replace The Frame's Page

Frames[Index] = Pages[i];

}

// Show Frames

for(int j=0;j<NFrames;j++)

{

cout<<Frames[j]<<" ";

}

cout<<endl;

} // End of Loop on Pages

// Show Tota Miss

cout<<"Total Miss = "<<TotalMiss<<endl;

}

// SecondChance function

void SecondChance(int Pages[],int NPages,int NFrames)

{

int \*frames = new int [NFrames] ; /\* array for frames \*/

bool \*secondChanceBit = new bool[NFrames] ; /\*SECOND CHANCE Bit \*/

bool valid[10];

int frame = 0; /\* index of the next frame to add pages in \*/

bool done; /\* check if page find frame \*/

int totalMiss = 0;

//initialize frames as empty

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

{

frames[i]= -1;

secondChanceBit[i] = false;

}

// initialize all valid with 0 for expected pages 1-10

for (int i=0; i<10; i++)

valid[i] = false;

for (int readyPage=0; readyPage<NPages; readyPage++)

{

do

{

if (frames[frame]==-1 && valid[Pages[readyPage]-1]== false)

{

cout<<"first condition";

frames[frame]=Pages[readyPage];

valid[Pages[readyPage]-1] = true;

secondChanceBit[frame] = false;

cout<<"you are at frame "<< frame;

frame = (frame+1)%NFrames;

}

else if (valid[Pages[readyPage]-1] == true)

{

cout<<"second condition";

cout<<"you are at frame "<< frame;

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

{

if (Pages[readyPage]==frames[i])

secondChanceBit[i]=true;

}

}

else if (secondChanceBit[frame]== true)

{

cout<<"third condition" <<"\n";

cout<<"you are at frame "<< frame;

secondChanceBit[frame] = false;

frame = (frame+1)%NFrames;

}

else if (secondChanceBit[frame]== false)

{

cout<<"fourth condition";

cout<<"you are at frame "<< frame;

valid[frames[frame]-1] = false;

frames[frame]=Pages[readyPage];

secondChanceBit[frame] = false;

frame = (frame+1)%NFrames;

valid[Pages[readyPage]-1] = true;

totalMiss++;

}

}

while (valid[Pages[readyPage]-1] == false);

cout << "total miss: "<<totalMiss<< "\n";

for (int qq = 0 ; qq<NFrames; qq++)

cout<< frames[qq]<<" ";

cout<< "\n";

}

}

// Main function

int main()

{

// Entering seed number

int seednumber;

cout << "Enter seed number: " ;

cin >> seednumber;

// Function to change time for Random function

srand (seednumber);

// Variables

int NPages; /\* TO Set The Number Of Pages \*/

int NFrames; /\* To Set The Number Of Frames \*/

int Algorithm; /\* To Chooses memory management algorithm\*/

// Taking The Numbers Of Pages and Frames

cout << "Enter The Number of Pages : ";

cin >> NPages ;

cout << "Enter The Number of Frames : ";

cin >> NFrames ;

system("CLS"); /\* Clearing the output screen \*/

// Pages Array

int \*Pages = new int [NPages];

// Pushing Random Numbers into the array from (1) to (10)

cout << "Array : "<<endl;

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

//cin >> Pages[i];

Pages[i] = rand() % 10 + 1;

cout << Pages[i]<<" ";

}

// For Choosing a Number

cout <<endl<<endl;

cout << "1- First In First Out(FIFO)"<<endl;

cout << "2- Least Recently used(LRU)"<<endl;

cout << "3- Least Frequently used(LFU)"<<endl;

cout << "4- Most Frequently used(MFU)"<<endl;

cout << "5- Optimal "<<endl;

cout << "6- Second Chance"<<endl<<endl;

cout << "Choose a Number : ";

// Taking an input

cin >> Algorithm;

// system("CLS"); /\* Clearing the output screen \*/

// Checking the input for executing the certain function

if (Algorithm == 1){

/\* Headers/FIFO.h \*/

FIFO(Pages , NPages, NFrames);

}

else if (Algorithm == 2){

/\* Headers/LRU.h \*/

LRU(Pages , NPages, NFrames);

}

else if (Algorithm == 3){

/\* Headers/LFU.h \*/

LFU(Pages , NPages, NFrames);

}

else if (Algorithm == 4){

/\* Headers/MFU.h \*/

MFU(Pages , NPages, NFrames);

}

else if (Algorithm == 5){

/\* Headers/Optimal.h \*/

Optimal(Pages , NPages, NFrames);

}

else if (Algorithm == 6){

/\* Headers/SecondChance.h \*/

SecondChance(Pages, NPages, NFrames);

}else{

cout << "Please Choose a valid Number"<<endl;

}

return 0;

}