#include<stdio.h>

#include<stdlib.h>

int pageQIndex = 0;

int pageFaults = 0; // Variable to count page faults

void initializeFrames(int maxframes, int frames[], int pageQ[])

{

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

{

frames[i] = -1; // Initialize frames with -1 indicating empty

pageQ[i] = -1; // Initialize page queue with -1

}

}

void displayFrames(int maxframes, int frames[])

{

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

{

if (frames[i] == -1)

{

printf("- ");

}

else

{

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

}

}

printf("\n");

}

int isPageInFrames(int page, int maxframes, int frames[])

{

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

{

if (frames[i] == page)

{

return 1; // Page found in frames

}

}

return 0; // Page not found in frames

}

int findEmptyFrame(int maxframes, int frames[])

{

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

{

if (frames[i] == -1)

{

return i; // Return the index of an empty frame

}

}

return -1; // No empty frame found

}

int findLRUFrame(int pageQ[])

{

return pageQ[0]; // LRU frame is the first page in the queue

}

int findOptimalFrame(int page, int Str[], int curridx, int maxframes, int maxpages, int frames[])

{

int farthestIndex = -1;

int maxDistance = -1;

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

{

int j;

for (j = curridx; j < maxpages; j++)

{

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

{

if (j > maxDistance)

{

maxDistance = j;

farthestIndex = i;

}

break;

}

}

if (j == maxpages)

{

return i; // Page not referenced again, so it's the best choice

}

}

return farthestIndex;

}

void FCFS(int Str[], int maxframes, int maxpages, int frames[], int pageQ[])

{

initializeFrames(maxframes, frames, pageQ);

pageFaults = 0;

printf("FCFS Page Replacement Algorithm:\n");

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

{

if (!isPageInFrames(Str[i],maxframes,frames))

{

int emptyFrameIndex = findEmptyFrame(maxframes,frames);

if (emptyFrameIndex != -1)

{

frames[emptyFrameIndex] = Str[i];

} else

{

frames[0] = Str[i];

}

pageFaults++;

}

displayFrames(maxframes,frames);

}

printf("Total Page Faults: %d\n", pageFaults);

}

void LRU(int Str[], int maxframes, int maxpages, int frames[], int pageQ[])

{

initializeFrames(maxframes, frames, pageQ);

pageFaults = 0;

printf("\nLRU Page Replacement Algorithm:\n");

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

{

if (!isPageInFrames(Str[i],maxframes,frames))

{

int emptyFrameIndex = findEmptyFrame(maxframes,frames);

if (emptyFrameIndex != -1)

{

frames[emptyFrameIndex] = Str[i];

}

else

{

int lruFrame = findLRUFrame(pageQ);

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

{

if (pageQ[j] == lruFrame)

{

frames[j] = Str[i];

break;

}

}

}

pageFaults++;

}

// Update the page queue

pageQ[pageQIndex] = Str[i];

pageQIndex++;

displayFrames(maxframes,frames);

}

printf("Total Page Faults: %d\n", pageFaults);

}

void Optimal(int Str[],int maxframes,int maxpages,int frames[],int pageQ[])

{

initializeFrames(maxframes,frames,pageQ);

pageFaults = 0;

printf("\nOptimal Page Replacement Algorithm:\n");

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

{

if (!isPageInFrames(Str[i],maxframes,frames))

{

int emptyFrameIndex = findEmptyFrame(maxframes,frames);

if (emptyFrameIndex != -1)

{

frames[emptyFrameIndex] = Str[i];

}

else

{

int optimalFrame = findOptimalFrame(Str[i], Str, i + 1,maxframes,maxpages,frames);

frames[optimalFrame] = Str[i];

}

pageFaults++;

}

displayFrames(maxframes,frames);

}

printf("Total Page Faults: %d\n", pageFaults);

}

int main()

{

int maxframes;

int maxpages;

printf("enter maxpages : ");

scanf("%d",&maxpages);

printf("enter maxframes : ");

scanf("%d",&maxframes);

int frames[maxframes];

int pageQ[maxframes];

int Str[maxpages];

printf("enter the number String: ");

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

{

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

}

//int Str[maxpages] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3};

FCFS(Str,maxframes,maxpages,frames,pageQ);

LRU(Str,maxframes,maxpages,frames,pageQ);

Optimal(Str,maxframes,maxpages,frames,pageQ);

return 0;

}

OUTPUT

