#include <stdio.h>

#include <stdbool.h>

#define MAX 20

bool isPageInFrame(int frames[], int n, int page)

{

    for (int i = 0; i < n; i++)

    {

        if (frames[i] == page)

            return true;

    }

    return false;

}

int FCFS(int pages[], int numPages, int frames[], int numFrames)

{

    int pageFaults = 0, pointer = 0;

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

        if (!isPageInFrame(frames, numFrames, pages[i]))

        {

            frames[pointer] = pages[i];

            pointer = (pointer + 1) % numFrames;

            pageFaults++;

        }

    }

    return pageFaults;

}

int LRU(int pages[], int numPages, int frames[], int numFrames)

{

    int pageFaults = 0, leastRecentlyUsed[MAX], time = 0;

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

    {

        time++;

        if (!isPageInFrame(frames, numFrames, pages[i]))

        {

            int lru = 0;

            for (int j = 1; j < numFrames; j++)

            {

                if (leastRecentlyUsed[j] < leastRecentlyUsed[lru])

                    lru = j;

            }

            frames[lru] = pages[i];

            pageFaults++;

        }

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

        {

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

            {

                leastRecentlyUsed[j] = time;

            }

        }

    }

    return pageFaults;

}

int Optimal(int pages[], int numPages, int frames[], int numFrames)

{

    int pageFaults = 0;

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

    {

        if (!isPageInFrame(frames, numFrames, pages[i]))

        {

            int farthest = -1, replace = 0;

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

            {

                int k;

                for (k = i + 1; k < numPages; k++)

                {

                    if (frames[j] == pages[k]) break;

                }

                if (k > farthest)

                {

                    farthest = k;

                    replace = j;

                }

            }

            frames[replace] = pages[i];

            pageFaults++;

        }

    }

    return pageFaults;

}

int main() {

    int pages[MAX], frames[MAX];

    int numPages, numFrames;

    printf("Enter number of pages: ");

    scanf("%d", &numPages);

    printf("Enter page reference sequence: ");

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

    {

        scanf("%d", &pages[i]);

    }

    printf("Enter number of frames (at least 3): ");

    scanf("%d", &numFrames);

    for (int i = 0; i < numFrames; i++) frames[i] = -1;

    printf("FCFS Page Faults: %d\n", FCFS(pages, numPages, frames, numFrames));

    for (int i = 0; i < numFrames; i++) frames[i] = -1;

    printf("LRU Page Faults: %d\n", LRU(pages, numPages, frames, numFrames));

    for (int i = 0; i < numFrames; i++) frames[i] = -1;

    printf("Optimal Page Faults: %d\n", Optimal(pages, numPages, frames, numFrames));

    return 0;

}

Test Case 1:

Input:

Number of Pages: 9

Page Reference Sequence: 1 3 0 3 5 6 3 2 1

Number of Frames: 3

Output:

FCFS Page Faults: 6

LRU Page Faults: 5

Optimal Page Faults: 4

Test Case 2:

Input:

Number of Pages: 12

Page Reference Sequence: 7 0 1 2 0 3 0 4 2 3 0 3

Number of Frames: 3

Output:

FCFS Page Faults: 9

LRU Page Faults: 7

Optimal Page Faults: 6

Test Case 3:

Input:

Number of Pages: 8

Page Reference Sequence: 2 3 2 1 5 2 4 5

Number of Frames: 3

Output:

FCFS Page Faults: 5

LRU Page Faults: 4

Optimal Page Faults: 4