1. Construct a C program to simulate the optimal paging technique of memory management

#include <stdio.h>

int isPresent(int frames[], int f, int page) {

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

if (frames[i] == page)

return 1;

}

return 0;

}

int predict(int pages[], int frames[], int n, int index, int f) {

int result = -1, farthest = index;

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

int j;

for (j = index; j < n; j++) {

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

if (j > farthest) {

farthest = j;

result = i;

}

break;

}

}

if (j == n)

return i;

}

return (result == -1) ? 0 : result;

}

int main() {

int pages[50], frames[10];

int n, f, i, j, pageFaults = 0, nextFree = 0;

printf("Enter number of pages: ");

scanf("%d", &n);

printf("Enter the page reference string:\n");

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

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

printf("Enter number of frames: ");

scanf("%d", &f);

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

frames[i] = -1;

printf("\nPage\tFrames\t\tPage Fault\n");

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

printf("%d\t", pages[i]);

if (isPresent(frames, f, pages[i])) {

// Page Hit

for (j = 0; j < f; j++) {

if (frames[j] != -1)

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

else

printf("- ");

}

printf("\t\tNo\n");

continue;

}

if (nextFree < f) {

frames[nextFree++] = pages[i];

} else {

int pos = predict(pages, frames, n, i + 1, f);

frames[pos] = pages[i];

}

pageFaults++;

for (j = 0; j < f; j++) {

if (frames[j] != -1)

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

else

printf("- ");

}

printf("\t\tYes\n");

}

printf("\nTotal Page Faults = %d\n", pageFaults);

return 0;

}