```
#include <stdio.h>
#include <stdlib.h>

#define MAX_PAGES 100
#define MAX_FRAMES 10

// Function to find the index of the farthest page in the future
int findFarthest(int pages[], int n, int index, int frames[], int frameSize) {
    int res = -1, farthest = index;
    for (int i = 0; i < frameSize; i++) {
        int j;
        for (j = index; j < n; j++) {
            if (frames[i] == pages[j]) {
                if (j > farthest) {
                    farthest = j;
                    res = i;
                }
                break;
            }
        }
        if (j == n)
            return i;
    }
    return (res == -1) ? 0 : res;
}

// First Come First Serve (FCFS) Page Replacement Algorithm
void fcfs(int pages[], int n, int frames[], int frameSize) {
    int faults = 0;
    int nextFrameIndex = 0;
    for (int i = 0; i < n; i++) {
```

```
        int j;
        for (j = 0; j < frameSize; j++) {
            if (frames[j] == pages[i]) {
                break;
            }
        }
        if (j == frameSize) {
            faults++;
            frames[nextFrameIndex] = pages[i];
            nextFrameIndex = (nextFrameIndex + 1) % frameSize;
        }
    }
    printf("FCFS Page Replacement Algorithm:\n");
    printf("Total Page Faults: %d\n", faults);
}

// Least Recently Used (LRU) Page Replacement Algorithm
void lru(int pages[], int n, int frames[], int frameSize) {
    int faults = 0;
    int time[MAX_FRAMES] = {0};
    for (int i = 0; i < n; i++) {
        int j;
        for (j = 0; j < frameSize; j++) {
            if (frames[j] == pages[i]) {
                time[j] = i + 1;
                break;
            }
        }
        if (j == frameSize) {
            faults++;
            int index = 0;
```

```
            for (int k = 1; k < frameSize; k++) {
                if (time[k] < time[index]) {
                    index = k;
                }
            }
            frames[index] = pages[i];
            time[index] = i + 1;
        }
    }
    printf("LRU Page Replacement Algorithm:\n");
    printf("Total Page Faults: %d\n", faults);
}

// Optimal Page Replacement Algorithm
void optimal(int pages[], int n, int frames[], int frameSize) {
    int faults = 0;
    for (int i = 0; i < n; i++) {
        int j;
        for (j = 0; j < frameSize; j++) {
            if (frames[j] == pages[i]) {
                break;
            }
        }
        if (j == frameSize) {
            faults++;
            int index = findFarthest(pages, n, i + 1, frames, frameSize);
            frames[index] = pages[i];
        }
    }
    printf("Optimal Page Replacement Algorithm:\n");
    printf("Total Page Faults: %d\n", faults);
```

```
}

int main() {
    int pages[MAX_PAGES], frames[MAX_FRAMES], n, frameSize;

    printf("Enter the number of pages: ");
    scanf("%d", &n);

    printf("Enter the page reference string: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &pages[i]);
    }

    printf("Enter the frame size (minimum 3): ");
    scanf("%d", &frameSize);

    printf("\n");

    fcfs(pages, n, frames, frameSize);
    lru(pages, n, frames, frameSize);
    optimal(pages, n, frames, frameSize);

    return 0;
}
```