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

#define MAX_PAGES 100
#define MAX_FRAMES 10

// Function to display the frames
void display_frames(int frames[], int m) {
    for (int i = 0; i < m; i++) {
        if (frames[i] != -1)
            printf("%d ", frames[i]);
        else
            printf("- ");
    }
    printf("\n");
}

// FIFO Page Replacement Algorithm
void fifo(int pages[], int n, int m) {
    int frames[m];
    int page_faults = 0, k = 0;

    // Initialize frames to -1 (empty)
    for (int i = 0; i < m; i++)
        frames[i] = -1;

    for (int i = 0; i < n; i++) {
        int page = pages[i];
        int found = 0;

        // Check if the page is already in the frames
        for (int j = 0; j < m; j++) {
            if (frames[j] == page) {
                found = 1;
                break;
            }
        }

        if (!found) {
            frames[k] = page;
            k = (k + 1) % m; // FIFO replacement
            page_faults++;
        }

        printf("Page: %d | Frames: ", page);
        display_frames(frames, m);
    }

    printf("Total page faults: %d\n", page_faults);
}

// LRU Page Replacement Algorithm
void lru(int pages[], int n, int m) {
    int frames[m];
    int page_faults = 0;
```

```
// Initialize frames to -1 (empty)
for (int i = 0; i < m; i++)
    frames[i] = -1;

for (int i = 0; i < n; i++) {
    int page = pages[i];
    int found = 0;

    // Check if the page is already in the frames
    for (int j = 0; j < m; j++) {
        if (frames[j] == page) {
            found = 1;
            break;
        }
    }

    if (!found) {
        // Find the least recently used page
        int lru = 0;
        for (int j = 1; j < m; j++) {
            if (frames[j] == -1) {
                lru = j;
                break;
            }
        }
        // Replace the least recently used page
        frames[lru] = page;
        page_faults++;
    }
```

```
        // Display the frames after each page access

        printf("Page: %d | Frames: ", page);

        display_frames(frames, m);

    }


    printf("Total page faults: %d\n", page_faults);

}


// Optimal Page Replacement Algorithm

void optimal(int pages[], int n, int m) {

    int frames[m];

    int page_faults = 0;


    // Initialize frames to -1 (empty)

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

        frames[i] = -1;


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

        int page = pages[i];

        int found = 0;


        // Check if the page is already in the frames

        for (int j = 0; j < m; j++) {

            if (frames[j] == page) {

                found = 1;

                break;

            }

        }
```

```
if (!found) {
    int farthest = -1, replace_index = -1;

    // Find the farthest page to replace
    for (int j = 0; j < m; j++) {
        int next_use = -1;
        for (int k = i + 1; k < n; k++) {
            if (frames[j] == pages[k]) {
                next_use = k;
                break;
            }
        }

        if (next_use == -1) {
            replace_index = j;
            break;
        } else {
            if (next_use > farthest) {
                farthest = next_use;
                replace_index = j;
            }
        }
    }

    // Replace the page
    frames[replace_index] = page;
    page_faults++;
}
```

```
        printf("Page: %d | Frames: ", page);
        display_frames(frames, m);
    }

    printf("Total page faults: %d\n", page_faults);
}

int main() {
    int pages[MAX_PAGES], n, m, choice;

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

    printf("Enter the number of frames: ");
    scanf("%d", &m);

    do {
        printf("\nPage Replacement Algorithms Menu:\n");
        printf("1. FIFO (First-In-First-Out)\n");
        printf("2. LRU (Least Recently Used)\n");
        printf("3. Optimal\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);


        switch (choice) {
            case 1:
                fifo(pages, n, m);
                break;
            case 2:
                lru(pages, n, m);
                break;
            case 3:
                optimal(pages, n, m);
                break;
            case 4:
                printf("Exiting program.\n");
                break;
            default:
                printf("Invalid choice! Please select again.\n");
        }
    } while (choice != 4);


    return 0;
}
```