NAME: G. GANESH

REG NO: 192373008

EXERICSE-33

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

AIM:

To construct a C program to simulate the Optimal Page Replacement technique used in memory management.

Algorithm:

1. Input: Number of frames, reference string length, and the reference string.
2. Initialization:
    - Initialize the frame array to -1 (indicating empty frames).
    - Set the page fault counter to 0.
3. Process Each Page in the Reference String:
    - If the page is already in the frame, continue to the next page (no page fault).
    - If the page is not in the frame:
        - Increment the page fault counter.
        - If there is an empty frame, place the page in the empty frame.
        - If all frames are occupied:
            - For each page in the frame, determine how far in the future it will be used.
            - Replace the page with the farthest future use (or not used at all) with the current page.
4. Output:
    - Display the reference string, frame status after each step, and the total number of page faults.

Procedure:

1. Read the reference string and number of frames.
2. Simulate the Optimal Page Replacement algorithm.
3. Display the output with the reference string, frames, and total page faults.

Code:

```
#include <stdio.h>
#define MAX 100
int find_farthest(int frames[], int ref[], int n, int index, int frame_count) {
    int farthest = -1, farthest_index = -1;
    for (int i = 0; i < frame_count; i++) {
        int found = 0;
        for (int j = index + 1; j < n; j++) {
            if (frames[i] == ref[j]) {
                if (j > farthest_index) {
                    farthest_index = j;
                    farthest = i;
                }
                found = 1;
                break;
            }
        }
        if (!found) return i;
    }
    return farthest;
}
int main() {
    int n, frame_count, ref[MAX], frames[MAX], page_faults = 0;
    printf("Enter the number of pages in the reference string: ");
    scanf("%d", &n);
    printf("Enter the reference string: ");
    for (int i = 0; i < n; i++) scanf("%d", &ref[i]);
    printf("Enter the number of frames: ");
    scanf("%d", &frame_count);
```

```
for (int i = 0; i < frame_count; i++) frames[i] = -1;
for (int i = 0; i < n; i++) {
    int found = 0;
    for (int j = 0; j < frame_count; j++) {
        if (frames[j] == ref[i]) {
            found = 1;
            break;
        }
    }
    if (!found) {
        page_faults++;
        int empty_index = -1;
        for (int j = 0; j < frame_count; j++) {
            if (frames[j] == -1) {
                empty_index = j;
                break;
            }
        }
        if (empty_index != -1) {
            frames[empty_index] = ref[i];
        } else {
            int replace_index = find_farthest(frames, ref, n, i, frame_count);
            frames[replace_index] = ref[i];
        }
    }
    printf("Step %d: ", i + 1);
    for (int j = 0; j < frame_count; j++) {
        if (frames[j] == -1) printf(" - ");
        else printf(" %d ", frames[j]);
    }
```

```
        printf("\n");
    }
    printf("Total Page Faults: %d\n", page_faults);
    return 0;
}
```

Result:

The program simulates the Optimal Page Replacement algorithm, displaying the frame status after each step and the total page faults.

Output: