NAME: G. GANESH

REG NO: 192373008

EXERICSE-31

Construct a C program to simulate the First in First Out paging technique of memory management.

AIM:

To implement and simulate the **First In First Out (FIFO)** paging technique in memory management using a C program.

Algorithm:

1. **Input:**
    - The number of pages in the reference string.
    - The reference string.
    - The number of frames available in memory.
2. **Initialize:**
    - A queue to hold pages currently in memory.
    - A counter to track page faults.
    - The front and rear pointers of the queue.
3. **Process each page in the reference string:**
    - If the page is already in memory, it is a page hit.
    - If the page is not in memory:
        - If there is space in the memory, add the page to the queue.
        - If memory is full, remove the page that was loaded first (FIFO) and add the new page to the queue.
        - Increment the page fault counter.
4. **Output:**
    - The number of page faults.
    - The sequence of pages in memory after each step.

Procedure:

1. Accept the reference string and number of frames as input.
2. Simulate the FIFO page replacement algorithm using a queue.
3. For each page, check if it is in memory:
    - If present, it is a hit.
    - Otherwise, replace the oldest page if memory is full.
4. Maintain a count of page faults and display the sequence of memory states.

Code:

```
#include <stdio.h>
#include <stdbool.h>
#define MAX 100
bool isPageInMemory(int memory[], int n, int page) {
    for (int i = 0; i < n; i++) {
        if (memory[i] == page) {
            return true;
        }
    }
    return false;
}
int main() {
    int n, frames, pageFaults = 0, current = 0;
    int reference[MAX], memory[MAX];
    printf("Enter the number of pages: ");
    scanf("%d", &n);
    printf("Enter the reference string: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &reference[i]);
    }
    printf("Enter the number of frames: ");
    scanf("%d", &frames);
```

```
for (int i = 0; i < frames; i++) {
    memory[i] = -1;
}
printf("\nPage Replacement Process:\n");
for (int i = 0; i < n; i++) {
    printf("Reference page: %d | Memory state: ", reference[i]);
    if (!isPageInMemory(memory, frames, reference[i])) {
        memory[current] = reference[i];
        current = (current + 1) % frames; // FIFO replacement
        pageFaults++;
        for (int j = 0; j < frames; j++) {
            if (memory[j] != -1) {
                printf("%d ", memory[j]);
            } else {
                printf("- ");
            }
        }
        printf(" <- Page Fault\n");
    } else {
        for (int j = 0; j < frames; j++) {
            if (memory[j] != -1) {
                printf("%d ", memory[j]);
            } else {
                printf("- ");
            }
        }
        printf(" <- Page Hit\n");
    }
}
printf("\nTotal Page Faults: %d\n", pageFaults);
```

```
    return 0;
}
```

Result:

The program simulates the FIFO page replacement technique. It tracks and displays:

- The memory state after each page reference.
- Whether each page reference causes a **page hit** or a **page fault**.
- The total number of page faults.

Output: