NAME: G. GANESH

REG NO: 192373008

EXERICSE-32

Construct a C program to simulate the Least Recently Used paging technique of memory management.

AIM:

To implement and simulate the **Least Recently Used (LRU)** 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:**
   - **An array to hold pages currently in memory.**
   - **A counter to track page faults.**
   - **An array to track the time of last use for each page.**
3. **Process each page in the reference string:**
   - **If the page is already in memory, it is a page hit, and update its last used time.**
   - **If the page is not in memory:**
     - **If there is space in memory, add the page to the memory.**
     - **If memory is full, replace the least recently used page.**
     - **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 LRU page replacement algorithm using an array and time counters.
3. For each page:
    - Check if it is in memory.
    - If it is not, find and replace the least recently used page.
4. Maintain a count of page faults and display the sequence of memory states.

Code:

```
#include <stdio.h>
#include <stdbool.h>
#include <limits.h>
#define MAX 100
int findLRU(int time[], int n) {
    int min = INT_MAX, pos = -1;
    for (int i = 0; i < n; i++) {
        if (time[i] < min) {
            min = time[i];
            pos = i;
        }
    }
    return pos;
}
int main() {
    int n, frames, pageFaults = 0, timeCounter = 0;
    int reference[MAX], memory[MAX], time[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;
    time[i] = 0;
}
printf("\nPage Replacement Process:\n");
for (int i = 0; i < n; i++) {
    printf("Reference page: %d | Memory state: ", reference[i]);
    bool found = false;
    for (int j = 0; j < frames; j++) {
        if (memory[j] == reference[i]) {
            found = true;
            time[j] = ++timeCounter; // Update the last used time
            break;
        }
    }
    if (!found) {
        int pos = -1;
        for (int j = 0; j < frames; j++) {
            if (memory[j] == -1) {
                pos = j;
                break;
            }
        }
        if (pos == -1) {
            pos = findLRU(time, frames);
        }
        memory[pos] = reference[i];
        time[pos] = ++timeCounter;
```

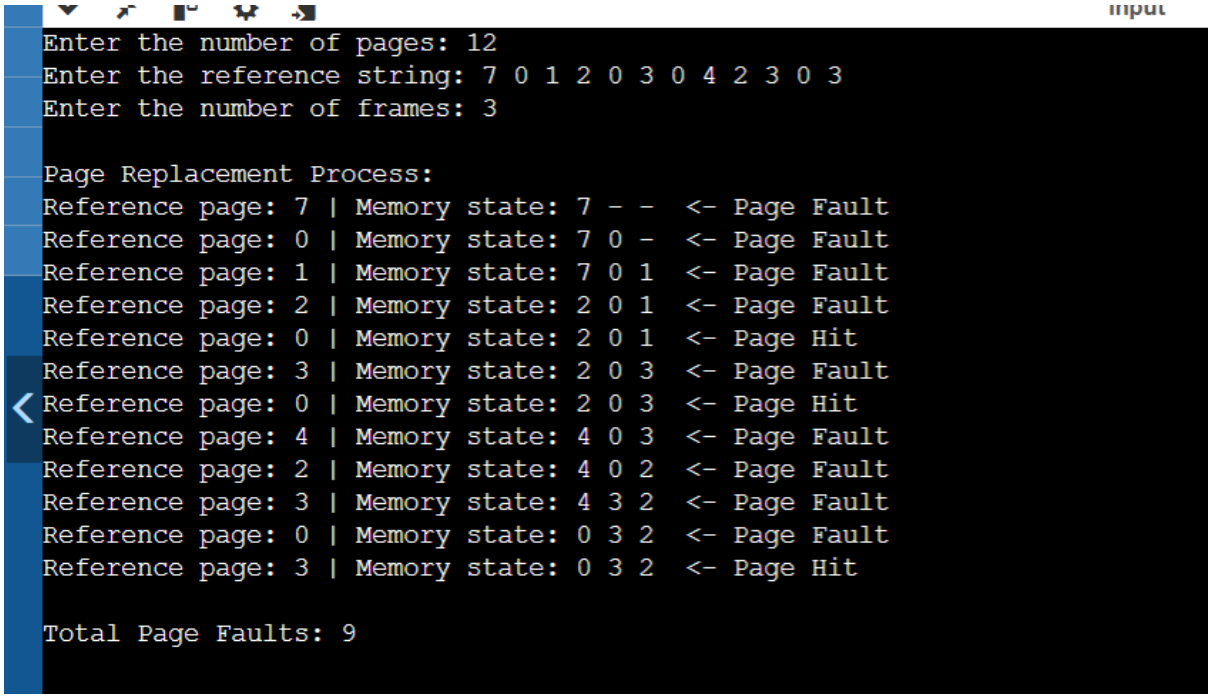
```
pageFaults++;
            for (int j = 0; j < frames; j++) {
                if (memory[j] != -1) {
                    printf("%d ", memory[j]);
                } else {
                    printf("- ");
                }
            }
            printf(" <- Page Fault\n");
        } else {
            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 **Least Recently Used (LRU)** 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:


Input
Enter the number of pages: 12
Enter the reference string: 7 0 1 2 0 3 0 4 2 3 0 3
Enter the number of frames: 3
Page Replacement Process:
Reference page: 7 | Memory state: 7 - - <- Page Fault
Reference page: 0 | Memory state: 7 0 - <- Page Fault
Reference page: 1 | Memory state: 7 0 1 <- Page Fault
Reference page: 2 | Memory state: 2 0 1 <- Page Fault
Reference page: 0 | Memory state: 2 0 1 <- Page Hit
Reference page: 3 | Memory state: 2 0 3 <- Page Fault
Reference page: 0 | Memory state: 2 0 3 <- Page Hit
Reference page: 4 | Memory state: 4 0 3 <- Page Fault
Reference page: 2 | Memory state: 4 0 2 <- Page Fault
Reference page: 3 | Memory state: 4 3 2 <- Page Fault
Reference page: 0 | Memory state: 0 3 2 <- Page Fault
Reference page: 3 | Memory state: 0 3 2 <- Page Hit
Total Page Faults: 9