192311035

M.RAVI SAI VINAY

EXPERIMENT-32

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

AIM:-

To simulate the Least Recently Used (LRU) page replacement algorithm and show how memory management works using this technique.

ALGORITHM:-

1. Input: Take the reference string (sequence of page requests) and the number of frames (available memory slots).
2. Page Fault: A page fault occurs when a requested page is not in memory.
3. FIFO Replacement: When memory is full and a page fault occurs, replace the oldest page (first inserted page) in memory.
4. Display: Print the page frames and the number of page faults.

PROCEDURE:-

1. Initialize the page frame as empty.
2. For each page request in the reference string:
    - If the page is not in memory, cause a page fault and insert the page into memory.
    - If memory is full, replace the least recently used page using LRU.
3. Display the page frames after each page request and count the number of page faults.

CODE:-

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

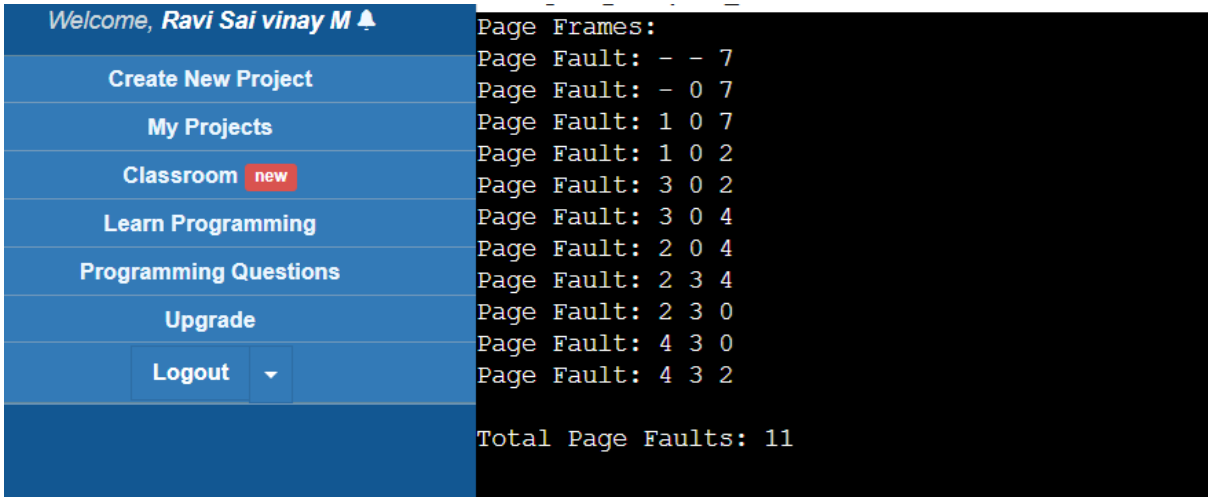
```
#define MAX_FRAMES 10

// Function to simulate LRU page replacement
void lruPageReplacement(int referenceString[], int numPages, int numFrames) {
    int frames[numFrames];  // Array to hold pages in memory
    int lastUsed[numFrames]; // Array to track last used time for each frame
    int pageFaults = 0;      // Count of page faults

    // Initialize frames as empty (-1 means empty) and lastUsed as 0
    for (int i = 0; i < numFrames; i++) {
        frames[i] = -1;
        lastUsed[i] = 0;
    }

    printf("Page Frames:\n");

    // Process each page in the reference string
    for (int i = 0; i < numPages; i++) {
        int page = referenceString[i];
        int pageFault = 1;
        int lruIndex = -1;

        // Check if the page is already in memory
        for (int j = 0; j < numFrames; j++) {
```

```
    if (frames[j] == page) {
        pageFault = 0;  // No page fault, page is already in memory
        lastUsed[j] = i;  // Update the last used time
        break;
    }
}

// If page is not in memory, cause a page fault
if (pageFault) {
    // Find the least recently used (LRU) page
    for (int j = 0; j < numFrames; j++) {
        if (frames[j] == -1 || lastUsed[j] < lastUsed[lruIndex]) {
            lruIndex = j;
        }
    }

    // Replace the LRU page with the new page
    frames[lruIndex] = page;
    lastUsed[lruIndex] = i;  // Update the last used time
    pageFaults++;

    // Print the current frame contents
    printf("Page Fault: ");
    for (int k = 0; k < numFrames; k++) {
```

```
            if (frames[k] == -1) {
                printf("- ");
            } else {
                printf("%d ", frames[k]);
            }
        }
        printf("\n");
    }
}

int main() {
    // Reference string (sequence of page requests)
    int referenceString[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 4, 2};
    int numPages = sizeof(referenceString) / sizeof(referenceString[0]);

    int numFrames = 3;  // Number of frames in memory

    // Simulate LRU page replacement
    lruPageReplacement(referenceString, numPages, numFrames);

    return 0;
```

}

OUTPUT:-



RESULT:-

- **LRU Page Replacement**: The program simulates the LRU page replacement technique correctly.
- **Page Faults**: It correctly identifies when a page fault occurs and replaces the least recently used page when necessary.
- **Output**: The program outputs the content of the page frames and the total number of page faults.