NAME:P.MALREDDY
REG:192372015

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

AIM

To construct a C program that simulates the **Least Recently Used (LRU)** paging technique of memory management, which replaces the page that has not been used for the longest time when a new page needs to be loaded and all frames are full.

PROCEDURE

1. **Start**
2. Input the total number of pages, the sequence of page references, and the number of available frames.
3. Initialize the frames as empty (-1), set the page fault counter to 0, and maintain an array to track usage timestamps of each frame.
4. For each page in the reference sequence:
    - Check if the page is already present in any of the frames.
        - If found, update its usage timestamp and move to the next page.
    - If not found:
        - If a frame is empty, load the page into the empty frame and update the timestamp.
        - If all frames are full, replace the page with the least recent usage timestamp with the current page.
        - Increment the page fault counter.
    - Display the current status of the frames.
5. Display the total number of page faults after processing all pages.
6. **Stop**

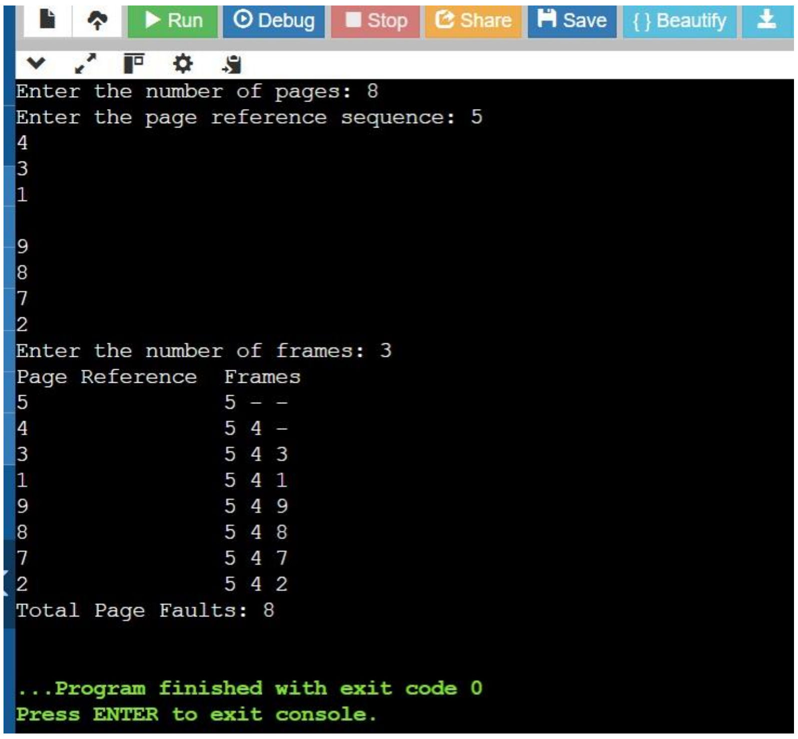
1. CODE:

```
#include <stdio.h>


void lruPaging(int pages[], int n, int frames[], int f) {
```

```
int pageFaults = 0, i, j, found, min, minIndex;

printf("Page Reference\tFrames\n");
for (i = 0; i < n; i++) {
  found = 0;
  for (j = 0; j < f; j++) {
    if (frames[j] == pages[i]) {
      found = 1;
      break;
    }
  }

  if (!found) {
    if (pageFaults < f) {
      frames[pageFaults] = pages[i];
    } else {
      min = 9999;
      for (j = 0; j < f; j++) {
        int usageCount = 0;
        for (int k = i - 1; k >= 0; k--) {
          if (pages[k] == frames[j]) {
            usageCount = i - k;
            break;
          }
        }
        if (usageCount < min) {
          min = usageCount;
          minIndex = j;
        }
      }
```

```
                frames[minIndex] = pages[i];
            }
            pageFaults++;
        }

        printf("%d\t\t", pages[i]);
        for (j = 0; j < f; j++) {
            if (frames[j] != -1) {
                printf("%d ", frames[j]);
            } else {
                printf("- ");
            }
        }
        printf("\n");
    }

    printf("Total Page Faults: %d\n", pageFaults);
}
int main() { int n, f, i;

    printf("Enter the number of pages: ");
    scanf("%d", Cn);

    int pages[n];
    printf("Enter the page reference sequence: ");
    for (i = 0; i < n; i++) {
        scanf("%d", Cpages[i]);
    }

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

    int frames[f];
    for (i = 0; i < f; i++) {
        frames[i] = -1;
    }

    lruPaging(pages, n, frames, f);

    return 0;
}
```

OUTPUT:



```
Enter the number of pages: 8
Enter the page reference sequence: 5
4
3
1

9
8
7
2
Enter the number of frames: 3
Page Reference  Frames
5               5 - -
4               5 4 -
3               5 4 3
1               5 4 1
9               5 4 9
8               5 4 8
7               5 4 7
2               5 4 2
Total Page Faults: 8


...Program finished with exit code 0
Press ENTER to exit console.
```