A Harsha Mukundha - 192324070

**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.

## ALGORITHM

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:

   o Check if the page is already present in any of the frames.

      ▪ If found, update its usage timestamp and move to the next page.

   o 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.

   o Display the current status of the frames.

5. Display the total number of page faults after processing all pages.

6. **Stop**

## PROCEDURE

1. Define the number of pages and frames.

2. Create an array for frames and initialize it as empty.

3. Create an array to track the last usage of pages.

4. Iterate over the page reference sequence, updating the frames based on the LRU replacement rule.

5. Display the current frame status and total page faults.

CODE:

```c
#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]) {
```

```c
                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);
}
```

```c
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: 9
Enter the page reference sequence: 1 3 0 3 5 6 3 1 3
Enter the number of frames: 3
Page Reference   Frames
1                1 - -
3                1 3 -
0                1 3 0
3                1 3 0
5                1 5 0
6                1 6 0
3                1 3 0
1                1 3 0
3                1 3 0
Total Page Faults: 6
```