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

A. program:

```
#include <stdio.h>

#include <stdbool.h>

#include <limits.h>


#define MAX_FRAMES 10   // Maximum number of frames

#define MAX_PAGES 50    // Maximum number of pages


// Function to find the least recently used page

int findLRU(int time[], int frames) {

   int min = INT_MAX, pos = -1;

   for (int i = 0; i < frames; i++) {

      if (time[i] < min) {

         min = time[i];

         pos = i;

      }

   }

   return pos;  // Return the position of the least recently used page

}


// Function to check if a page is already in memory

bool isPageInMemory(int memory[], int frames, int page, int *pos) {

   for (int i = 0; i < frames; i++) {

      if (memory[i] == page) {

         *pos = i;

         return true;  // Page is in memory
```

```
        }
    }
    return false;  // Page is not in memory
}

int main() {
    int frames;  // Number of frames
    int pages[MAX_PAGES];  // Reference string
    int memory[MAX_FRAMES];  // Memory to hold pages
    int time[MAX_FRAMES];  // Array to store time counters for LRU
    int pageCount, pageFaults = 0, clock = 0;

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

    printf("Enter the number of pages in the reference string: ");
    scanf("%d", &pageCount);

    printf("Enter the reference string: ");
    for (int i = 0; i < pageCount; i++) {
        scanf("%d", &pages[i]);
    }

    // Initialize memory and time arrays
    for (int i = 0; i < frames; i++) {
        memory[i] = -1;  // Empty memory
        time[i] = 0;     // No time recorded initially
```

```
}

for (int i = 0; i < pageCount; i++) {
    int currentPage = pages[i];
    int pos = -1;

    // Check if the page is already in memory
    if (isPageInMemory(memory, frames, currentPage, &pos)) {
        // Page hit: Update the time for the accessed page
        time[pos] = clock;
    } else {
        // Page fault occurs
        pageFaults++;

        // Find the least recently used page if memory is full
        if (i < frames) {
            pos = i;  // Fill empty memory slots first
        } else {
            pos = findLRU(time, frames);
        }

        // Replace the LRU page with the current page
        memory[pos] = currentPage;
        time[pos] = clock;
    }

    clock++;  // Increment the clock
```

```
        // Display the current memory state
        printf("Step %d: Memory: ", i + 1);
        for (int j = 0; j < frames; j++) {
            if (memory[j] == -1)
                printf(" - ");  // Empty frame
            else
                printf(" %d ", memory[j]);
        }
        printf("\n");
    }

    printf("Total Page Faults: %d\n", pageFaults);

    return 0;
}
```

Output:

```
Enter the number of frames: 4
Enter the number of pages in the reference string: 9
Enter the reference string: 1
2
3
4
5
6
7
8
9
Step 1: Memory:  1  -  -  -
Step 2: Memory:  1  2  -  -
Step 3: Memory:  1  2  3  -
Step 4: Memory:  1  2  3  4
Step 5: Memory:  5  2  3  4
Step 6: Memory:  5  6  3  4
Step 7: Memory:  5  6  7  4
Step 8: Memory:  5  6  7  8
Step 9: Memory:  9  6  7  8
Total Page Faults: 9

--------------------------------
```