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

```
#include <stdio.h>

#define MAX_FRAMES 3// Number of frames in memory
#define MAX_PAGES 20// Number of pages to be referenced

// Function to simulate LRU page replacement algorithm
void lruPageReplacement(int pages[], int n, int frames) {
    int memory[frames];  // Array to store frames in memory
    int time[frames]; // Array to track the time of last usage for each page
    int page_faults = 0; // Number of page faults
    int current_time = 0; // Time counter to simulate page accesses

    // Initialize memory frames and time array
    for (int i = 0; i < frames; i++) {
        memory[i] = -1;
        time[i] = -1;
    }

    // Traverse through the page reference string
    for (int i = 0; i < n; i++) {
        int page = pages[i];
        int found = 0;

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

```
            found = 1;  // Page is found in memory
            time[j] = current_time; // Update time for the page
            break;
        }
    }

    // If the page is not found in memory, it's a page fault
    if (!found) {
        int lru_index = 0;

        // Find the Least Recently Used page (the one with the smallest time)
        for (int j = 1; j < frames; j++) {
            if (time[j] < time[lru_index]) {
                lru_index = j;
            }
        }

        // Replace the LRU page with the new page
        memory[lru_index] = page;
        time[lru_index] = current_time; // Update time for the new page

        page_faults++; // Increment page fault count

        // Print the current memory status
        printf("Page %d caused a page fault. Replacing with: ", page);
        for (int k = 0; k < frames; k++) {
            if (memory[k] != -1) {
                printf("%d ", memory[k]);
            }
```

```
            }
            printf("\n");
        }

        // Increment the time counter for the next page reference
        current_time++;
    }

    // Final output
    printf("\nTotal Page Faults: %d\n", page_faults);
}

int main() {
    int pages[MAX_PAGES] = {1,2,3,4,2,5,3,4,2,6,7,8,7,9,7,8,2,5,4,9}; // Reference string
    int frames = MAX_FRAMES;

    printf("Page reference string: ");
    for (int i = 0; i < MAX_PAGES; i++) {
        printf("%d ", pages[i]);
    }
    printf("\n");

    // Call the LRU page replacement function
    lruPageReplacement(pages, MAX_PAGES, frames);

    return 0;
}
```

Sample Output

```
Page reference string: 1 2 3 4 2 5 3 4 2 6 7 8 7 9 7 8 2 5 4 9
Page 1 caused a page fault. Replacing with: 1
Page 2 caused a page fault. Replacing with: 1 2
Page 3 caused a page fault. Replacing with: 1 2 3
Page 4 caused a page fault. Replacing with: 4 2 3
Page 5 caused a page fault. Replacing with: 4 2 5
Page 3 caused a page fault. Replacing with: 3 2 5
Page 4 caused a page fault. Replacing with: 3 4 5
Page 2 caused a page fault. Replacing with: 3 4 2
Page 6 caused a page fault. Replacing with: 6 4 2
Page 7 caused a page fault. Replacing with: 6 7 2
Page 8 caused a page fault. Replacing with: 6 7 8
Page 9 caused a page fault. Replacing with: 9 7 8
Page 2 caused a page fault. Replacing with: 2 7 8
Page 5 caused a page fault. Replacing with: 2 5 8
Page 4 caused a page fault. Replacing with: 2 5 4
Page 9 caused a page fault. Replacing with: 9 5 4

Total Page Faults: 16

--------------------------------
Process exited after 2.836 seconds with return value 0
Press any key to continue . . .
```