33.Construct a C program to simulate the optimal 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 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;
      }
   }
   return false;
}

// Function to find the optimal page to replace
int findOptimalPage(int memory[], int frames, int pages[], int pageIndex, int pageCount) {
   int farthest = pageIndex, pos = -1;

   for (int i = 0; i < frames; i++) {
      int j;
      // Find the next occurrence of the page in memory
```

```
        for (j = pageIndex + 1; j < pageCount; j++) {
            if (memory[i] == pages[j]) {
                if (j > farthest) {
                    farthest = j;
                    pos = i;
                }
                break;
            }
        }
        // If the page is not found in the future
        if (j == pageCount) {
            return i;
        }
    }
    return (pos == -1) ? 0 : pos;
}

int main() {
    int frames;  // Number of frames
    int pages[MAX_PAGES];  // Reference string
    int memory[MAX_FRAMES];  // Memory to hold pages
    int pageCount, pageFaults = 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 to -1 (empty)
for (int i = 0; i < frames; i++) {
   memory[i] = -1;
}

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 fault occurs
      pageFaults++;

      if (i < frames) {
         // Fill empty slots in memory first
         memory[i] = currentPage;
      } else {
         // Find the optimal page to replace
         int replaceIndex = findOptimalPage(memory, frames, pages, i, pageCount);
```

```
            memory[replaceIndex] = currentPage;
        }
    }

    // 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
2
3
1
7
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:  1  2  3  5
Step 6: Memory:  1  2  3  5
Step 7: Memory:  1  2  3  5
Step 8: Memory:  1  2  3  5
Step 9: Memory:  7  2  3  5
Total Page Faults: 6

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