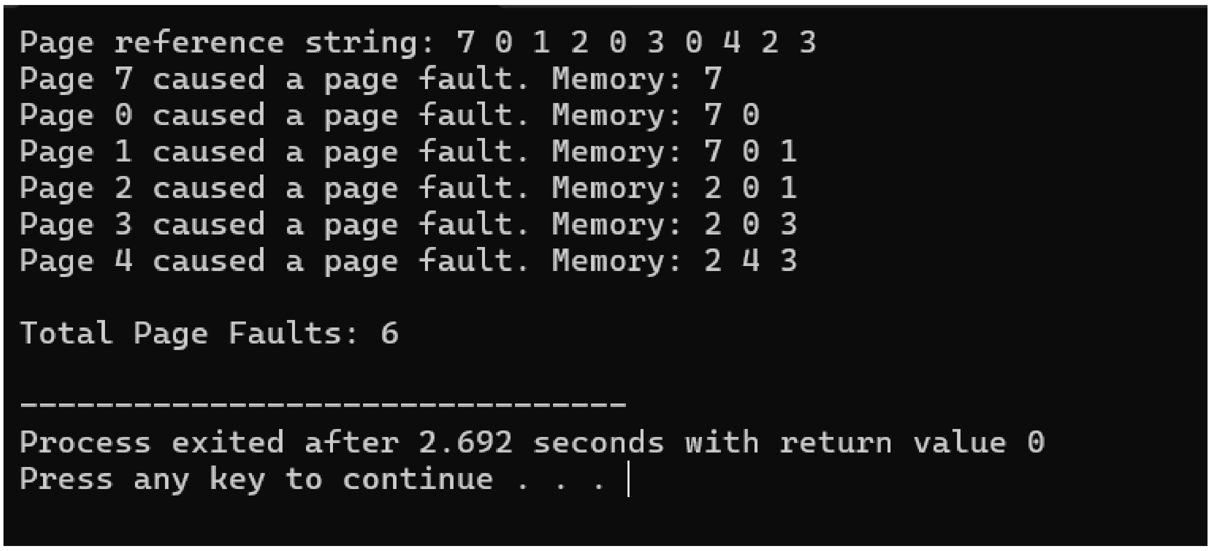
EXP 33: Construct a C program to simulate the optimal paging technique of memory management

```
#include <stdio.h>


#define MAX_FRAMES 3

#define MAX_PAGES 10


// Function to find the page to replace using Optimal strategy

int findOptimalIndex(int pages[], int memory[], int start, int n, int frames) {

    int index = -1, farthest = start;


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

        int j;

        for (j = start; j < n; j++) {

            if (memory[i] == pages[j]) {

                if (j > farthest) {

                    farthest = j;

                    index = i;

                }

                break;

            }

        }

        // If a page is never used again, replace it

        if (j == n) return i;

    }


    // If all pages are used again, replace the farthest used

    return (index == -1) ? 0 : index;
```

```
}

// Optimal page replacement simulation
void optimalPageReplacement(int pages[], int n, int frames) {
    int memory[frames];
    int page_faults = 0;
    int filled = 0;

    // Initialize memory frames
    for (int i = 0; i < frames; i++)
        memory[i] = -1;

    // Process each page
    for (int i = 0; i < n; i++) {
        int page = pages[i];
        int found = 0;

        // Check if page is already in memory
        for (int j = 0; j < frames; j++) {
            if (memory[j] == page) {
                found = 1;
                break;
            }
        }

        // If page is not found (page fault)
        if (!found) {
            if (filled < frames) {
                memory[filled++] = page;
            } else {
                int replaceIndex = findOptimalIndex(pages, memory, i + 1, n, frames);
                memory[replaceIndex] = page;
            }
            page_faults++;

            // Print current memory status
            printf("Page %d caused a page fault. Memory: ", page);
            for (int k = 0; k < frames; k++) {
                if (memory[k] != -1)
                    printf("%d ", memory[k]);
            }
            printf("\n");
        }
    }
    printf("\nTotal Page Faults: %d\n", page_faults);
}
int main() {
    int pages[MAX_PAGES] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3};
    int frames = MAX_FRAMES;

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

    return 0;
}
```

Sample Output



```
Page reference string: 7 0 1 2 0 3 0 4 2 3
Page 7 caused a page fault. Memory: 7
Page 0 caused a page fault. Memory: 7 0
Page 1 caused a page fault. Memory: 7 0 1
Page 2 caused a page fault. Memory: 2 0 1
Page 3 caused a page fault. Memory: 2 0 3
Page 4 caused a page fault. Memory: 2 4 3

Total Page Faults: 6

--------------------------------
Process exited after 2.692 seconds with return value 0
Press any key to continue . . .
```