# Green University of Bangladesh
## Department of Computer Science and Engineering(CSE)

**Faculty of Sciences and Engineering**
**Semester: (Summer, Year:2025), B.Sc. in CSE (Day)**

# Lab Report 03

**Course Title: Operating System Lab**
**Course Code: CSE 310  Section: 223 D2**

**Student Details**

| Name | ID |
|---|---|
| Anisur Rahaman Maruf | 222902078 |

**Submission Date: 19/08/2025**
**Course Teacher's Name: Umme Habiba**

**[For Teachers use only: Don't Write Anything inside this box]**

**Lab Report Status**

**Marks: …………………………………**  **Signature:.....................**
**Comments:.............................................**  **Date:..............................**

**Title:** Simulation of Optimal Page Replacement in Memory Management

## 1. Introduction

This program simulates the Optimal Page Replacement Algorithm. In operating systems, when memory is limited, old pages must be replaced to load new pages. The Optimal method replaces the page that will be used farthest in the future or will not be used again, resulting in the minimum number of page faults.

## 2. Objective

The objective of this program is to implement the Optimal Page Replacement Algorithm and calculate the number of page faults. This helps students understand page replacement strategies and the advantages of using the Optimal method in memory management.

## 3. Implementation

The program first takes the page sequence and the number of frames as input. If memory has empty slots, pages are loaded directly. Once memory is full, the program identifies the page that will be used farthest in the future or not used at all and replaces it with the new page. The findOptimal() function determines the index of the page to be replaced. At each step, the current state of memory and the page fault number are displayed.

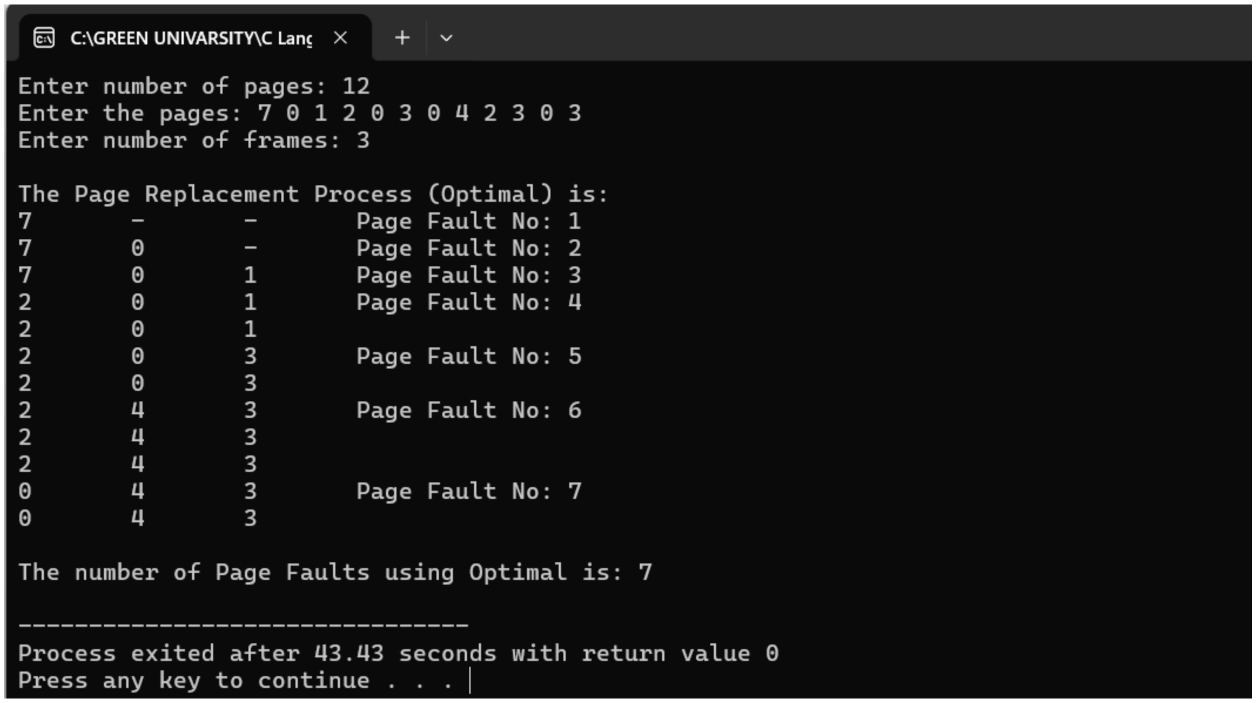
## Code

```
#include <stdio.h>

int findOptimal(int pages[], int memory[], int nf, int pos, int np) {
    int farthest = -1, idx = -1;
    for (int i = 0; i < nf; i++) {
        int j;
        for (j = pos + 1; j < np; j++) {
            if (memory[i] == pages[j]) {
                if (j > farthest) {
                    farthest = j;
                    idx = i;
                }
                break;
            }
        }
        if (j == np) return i;
    }
    return (idx == -1) ? 0 : idx;
}

int main() {
    int pf = 0, pages[50], mem[20];
    int np, nf, i, j;

    printf("Enter number of pages: ");
    scanf("%d", &np);
    printf("Enter the pages: ");
    for (i = 0; i < np; i++) scanf("%d", &pages[i]);
    printf("Enter number of frames: ");
    scanf("%d", &nf);
    for (i = 0; i < nf; i++) mem[i] = -1;

    printf("\nThe Page Replacement Process (Optimal) is:\n");
    for (i = 0; i < np; i++) {
        int found = 0;
        for (j = 0; j < nf; j++) {
            if (mem[j] == pages[i]) {
                found = 1;
                break;
            }
        }
        if (!found) {
            int idx = -1;
            for (j = 0; j < nf; j++) {
                if (mem[j] == -1) {
                    idx = j;
                    break;
                }
            }
            if (idx == -1) idx = findOptimal(pages, mem, nf, i, np);
            mem[idx] = pages[i];
            pf++;
        }
        for (j = 0; j < nf; j++) {
            if (mem[j] != -1) printf("%d\t", mem[j]);
            else printf("-\t");
        }
        if (!found) printf("Page Fault No: %d", pf);
        printf("\n");
    }
    printf("\nThe number of Page Faults using Optimal is: %d\n", pf);
    return 0;
}
```

## 4. Output

Enter number of pages: 12
Enter the pages: 7 0 1 2 0 3 0 4 2 3 0 3
Enter number of frames: 3

**Answer**:
The number of Page Faults using Optimal is: 9



```
Enter number of pages: 12
Enter the pages: 7 0 1 2 0 3 0 4 2 3 0 3
Enter number of frames: 3

The Page Replacement Process (Optimal) is:
7       -       -       Page Fault No: 1
7       0       -       Page Fault No: 2
7       0       1       Page Fault No: 3
2       0       1       Page Fault No: 4
2       0       1
2       0       3       Page Fault No: 5
2       0       3
2       4       3       Page Fault No: 6
2       4       3
2       4       3
0       4       3       Page Fault No: 7
0       4       3

The number of Page Faults using Optimal is: 7

--------------------------------
Process exited after 43.43 seconds with return value 0
Press any key to continue . . .
```

## 5. Discussion

From this program, it is clear that the Optimal algorithm results in fewer page faults compared to other methods because it makes replacement decisions based on future page references. However, in real systems, it is not always possible to know future references, which makes this method more theoretical than practical. When compared to FIFO or LRU, Optimal always produces the best possible result, but it is harder to implement in real-world scenarios.