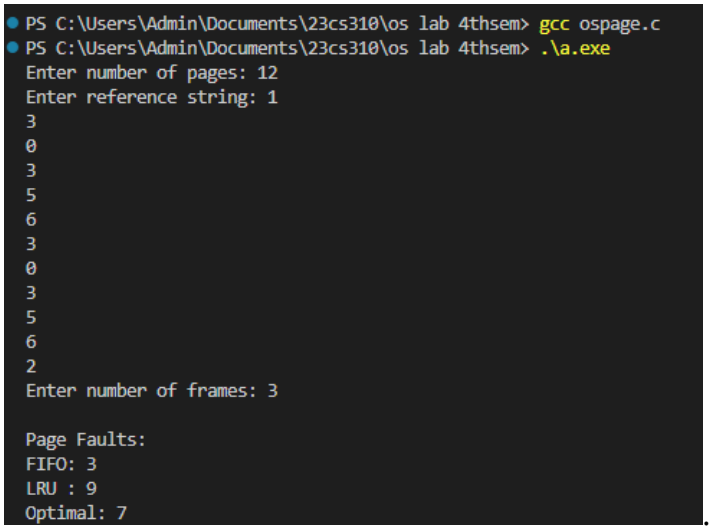
OS LAB – 15/05/2025

1. Write a c program to stimulate page replacement algorithms
   a) FIFO
   b) LRU
   c) Optimal

```
#include <stdio.h>
#define MAX 100

int inFrame(int frames[], int size, int page) {
    for (int i = 0; i < size; i++)
        if (frames[i] == page) return i;
    return -1;
}

int fifo(int pages[], int n, int cap) {
    int frames[MAX], index = 0, count = 0;
    for (int i = 0; i < n; i++) {
        if (inFrame(frames, count, pages[i]) == -1) {
            if (count < cap) frames[count++] = pages[i];
            else frames[index] = pages[i], index = (index + 1) % cap;
            count += (count < cap); // avoid double increment
        }
    }
    return count;
}

int lru(int pages[], int n, int cap) {
    int frames[MAX], recent[MAX], count = 0, faults = 0;
    for (int i = 0; i < n; i++) {
        int idx = inFrame(frames, count, pages[i]);
        if (idx != -1) recent[idx] = i;
        else {
            if (count < cap) {
                frames[count] = pages[i];
                recent[count++] = i;
            } else {
                int lru = 0;
                for (int j = 1; j < cap; j++)
                    if (recent[j] < recent[lru]) lru = j;
                frames[lru] = pages[i];
                recent[lru] = i;
            }
            faults++;
        }
    }
    return faults;
}

int optimal(int pages[], int n, int cap) {
    int frames[MAX], count = 0, faults = 0;
```

```
    for (int i = 0; i < n; i++) {
        if (inFrame(frames, count, pages[i]) != -1) continue;
        if (count < cap) frames[count++] = pages[i];
        else {
            int far = -1, idx = -1;
            for (int j = 0; j < cap; j++) {
                int k;
                for (k = i + 1; k < n; k++)
                    if (frames[j] == pages[k]) break;
                if (k > far) far = k, idx = j;
            }
            frames[idx] = pages[i];
        }
        faults++;
    }
    return faults;
}

int main() {
    int pages[MAX], n, cap;
    printf("Enter number of pages: ");
    scanf("%d", &n);
    printf("Enter reference string: ");
    for (int i = 0; i < n; i++) scanf("%d", &pages[i]);
    printf("Enter number of frames: ");
    scanf("%d", &cap);

    printf("\nPage Faults:\n");
    printf("FIFO: %d\n", fifo(pages, n, cap));
    printf("LRU : %d\n", lru(pages, n, cap));
    printf("Optimal: %d\n", optimal(pages, n, cap));
    return 0;
}
```

OUTPUT:



2. write a c program to stimulate the following file allocation strategies
   a) sequential
   b) indexed
   c) linked

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_BLOCKS 10

void sequentialAllocation(int blocks[], int total) {
    printf("Sequential: ");
    for (int i = 0; i < total; i++) printf("%d -> ", blocks[i]);
    printf("End\n");
}

void indexedAllocation(int index, int blocks[], int total) {
    printf("Indexed: Index Block %d -> [ ", index);
    for (int i = 0; i < total; i++) printf("%d ", blocks[i]);
    printf("]\n");
}

void linkedAllocation(int blocks[], int total) {
    printf("Linked: ");
    for (int i = 0; i < total; i++) printf("%d -> ", blocks[i]);
    printf("End\n");
}

int main() {
    int blocks[MAX_BLOCKS], total, choice, indexBlock;

    printf("Enter number of blocks (<=10): ");
```

```
    scanf("%d", &total);
    printf("Enter block numbers: ");
    for (int i = 0; i < total; i++) scanf("%d", &blocks[i]);
    printf("Choose allocation: 1. Sequential  2. Indexed  3. Linked\n");
    scanf("%d", &choice);

    if (choice == 2) {
        printf("Enter index block number: ");
        scanf("%d", &indexBlock);
    }

    if (choice == 1) sequentialAllocation(blocks, total);
    else if (choice == 2) indexedAllocation(indexBlock, blocks, total);
    else if (choice == 3) linkedAllocation(blocks, total);
    else printf("Invalid choice!\n");

    return 0;
}
```

OUTPUT: