Ex. No. : 11a **Date : 12.04.2025**

Register No. : 230701385 **Name : VISHWAK S**

FIFO

Aim:

To find out the number of page faults that occur using First-in First-out (FIFO) page replacement technique.

Algorithm:

1. Declare the size with respect to page length

2. Check the need of replacement from the page to memory

3. Check the need for replacement from old page to new page in memory

4. Form a queue to hold all pages

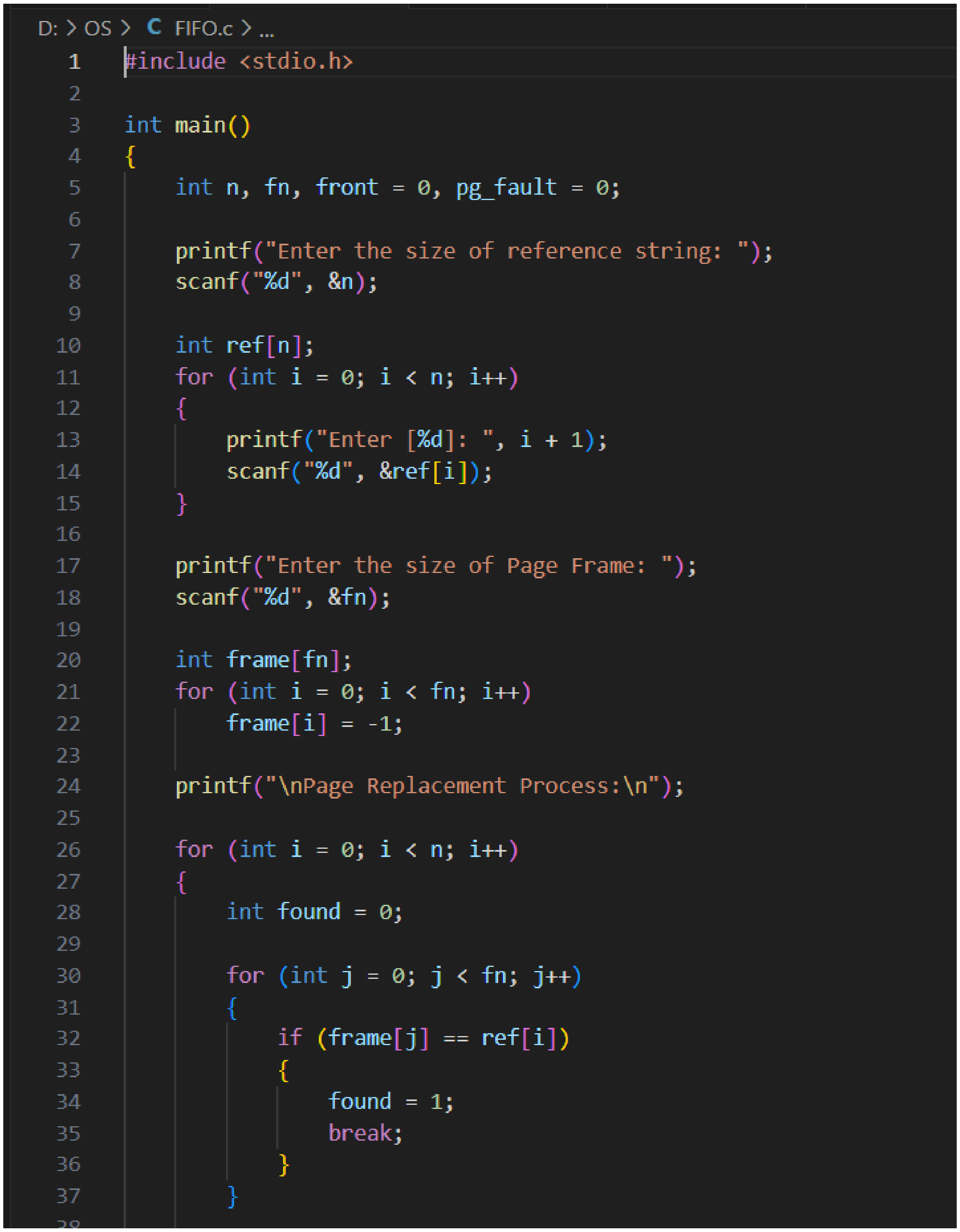
5. Insert the page require memory into the queue

6. Check for bad replacement and page fault

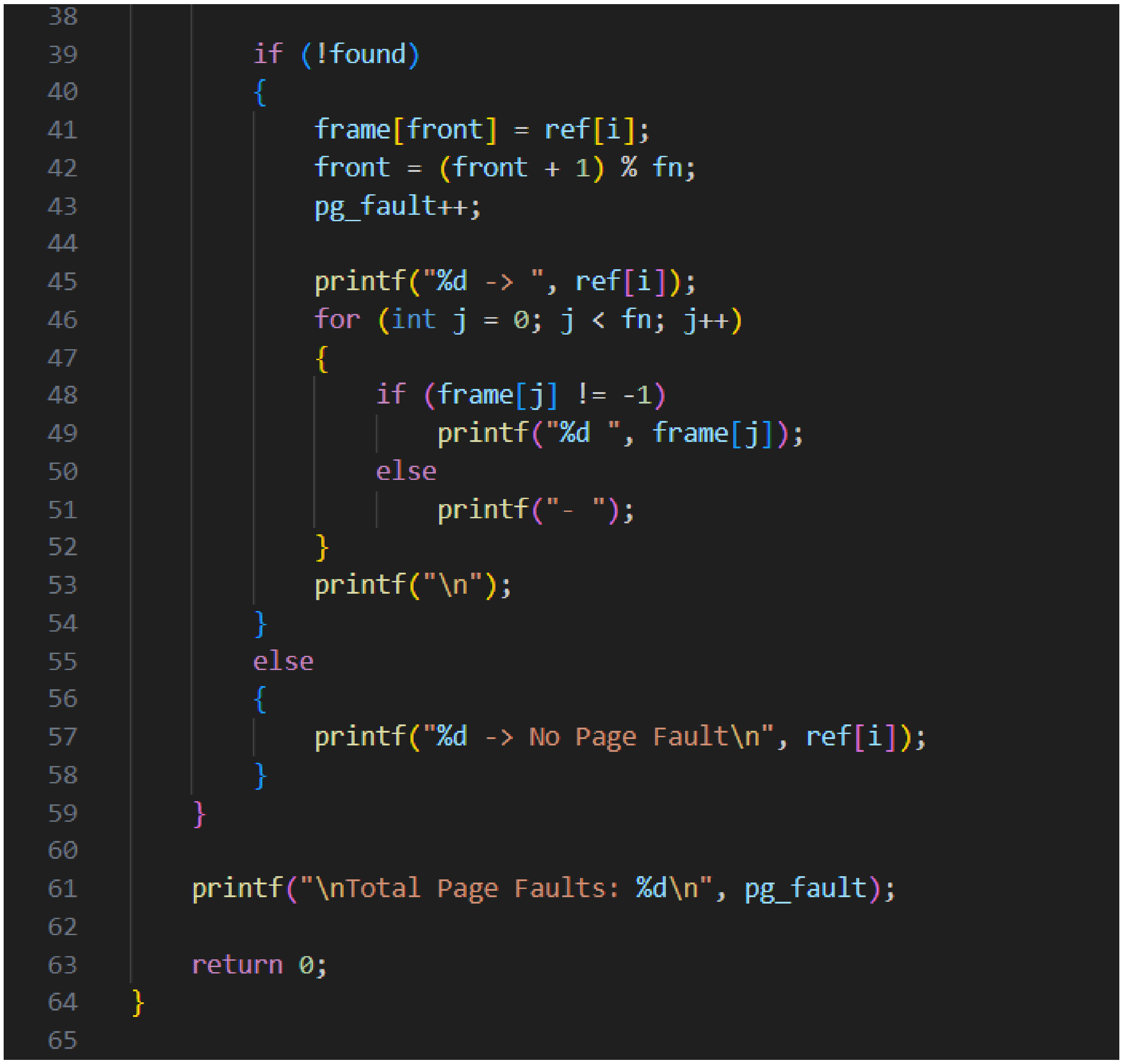
7. Get the number of processes to be inserted

8. Display the values.

Program:

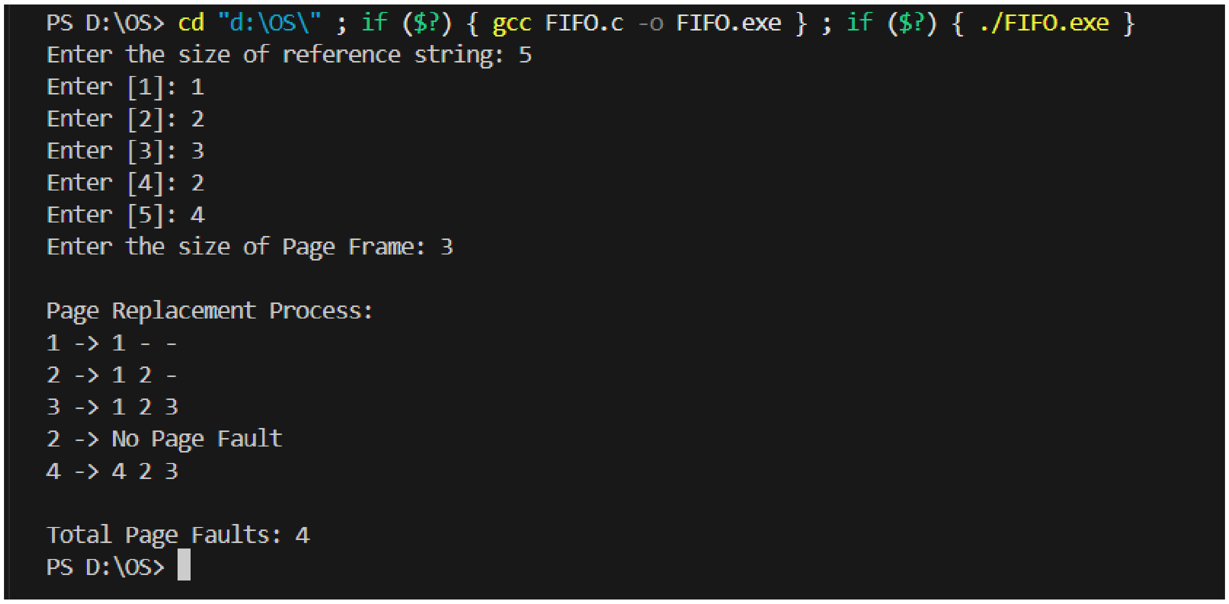

D: > OS > C FIFO.c > ...

```c
#include <stdio.h>

int main()
{
    int n, fn, front = 0, pg_fault = 0;

    printf("Enter the size of reference string: ");
    scanf("%d", &n);

    int ref[n];
    for (int i = 0; i < n; i++)
    {
        printf("Enter [%d]: ", i + 1);
        scanf("%d", &ref[i]);
    }

    printf("Enter the size of Page Frame: ");
    scanf("%d", &fn);

    int frame[fn];
    for (int i = 0; i < fn; i++)
        frame[i] = -1;

    printf("\nPage Replacement Process:\n");

    for (int i = 0; i < n; i++)
    {
        int found = 0;

        for (int j = 0; j < fn; j++)
        {
            if (frame[j] == ref[i])
            {
                found = 1;
                break;
            }
        }
```



```c

        if (!found)
        {
            frame[front] = ref[i];
            front = (front + 1) % fn;
            pg_fault++;

            printf("%d -> ", ref[i]);
            for (int j = 0; j < fn; j++)
            {
                if (frame[j] != -1)
                    printf("%d ", frame[j]);
                else
                    printf("- ");
            }
            printf("\n");
        }
        else
        {
            printf("%d -> No Page Fault\n", ref[i]);
        }
    }

    printf("\nTotal Page Faults: %d\n", pg_fault);

    return 0;
}

```

Output:



```
PS D:\OS> cd "d:\OS\" ; if ($?) { gcc FIFO.c -o FIFO.exe } ; if ($?) { ./FIFO.exe }
Enter the size of reference string: 5
Enter [1]: 1
Enter [2]: 2
Enter [3]: 3
Enter [4]: 2
Enter [5]: 4
Enter the size of Page Frame: 3

Page Replacement Process:
1 -> 1 - -
2 -> 1 2 -
3 -> 1 2 3
2 -> No Page Fault
4 -> 4 2 3

Total Page Faults: 4
PS D:\OS>
```

Result:

Hence the number of page faults that occur using First-in First-out (FIFO) page replacement technique has been found successfully.

Ex. No. : 11b **Date : 18.04.2025**

Register No. : 230701385 **Name : VISHWAK S**

LRU

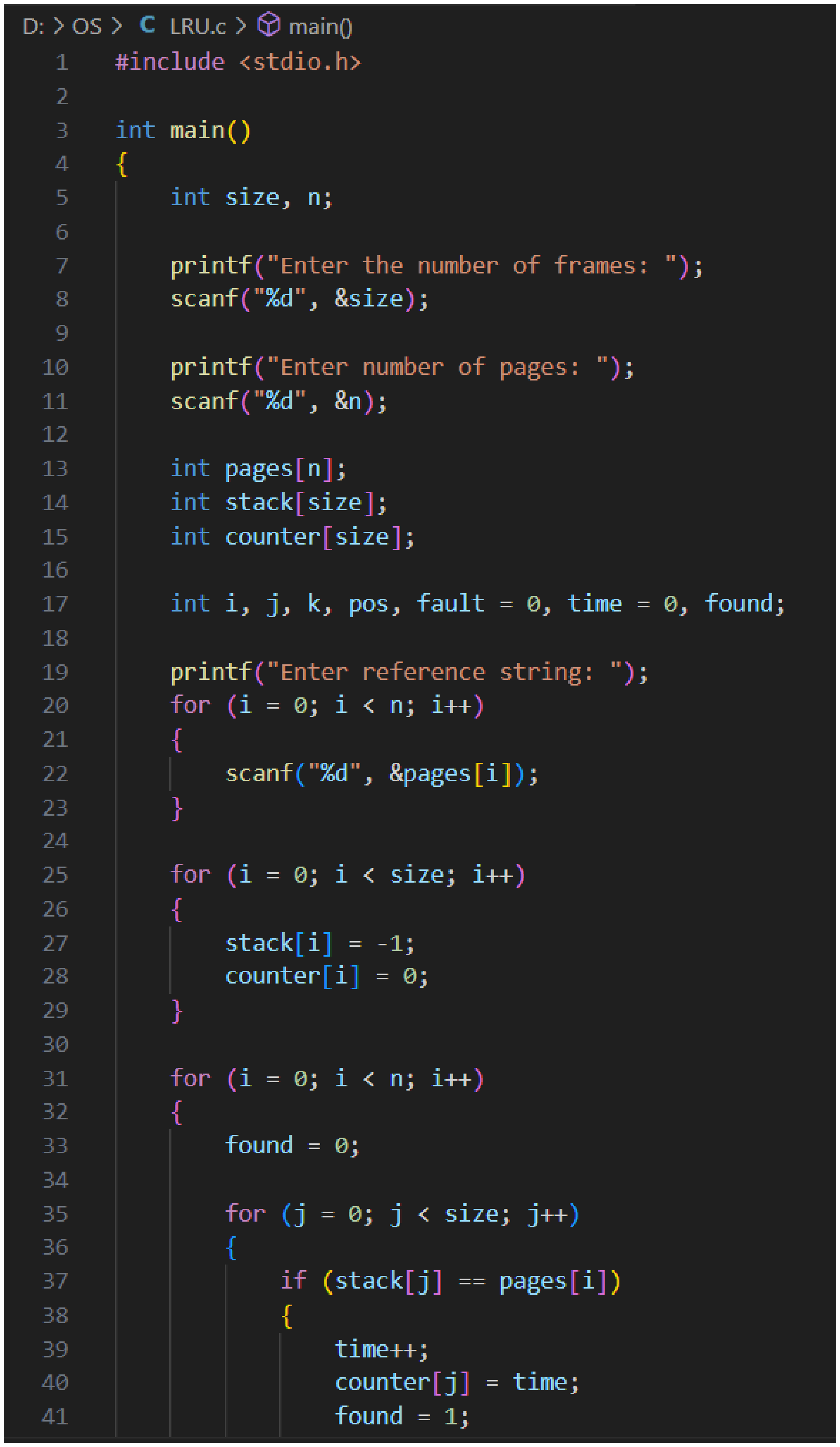
Aim:

To write a c program to implement LRU page replacement algorithm.

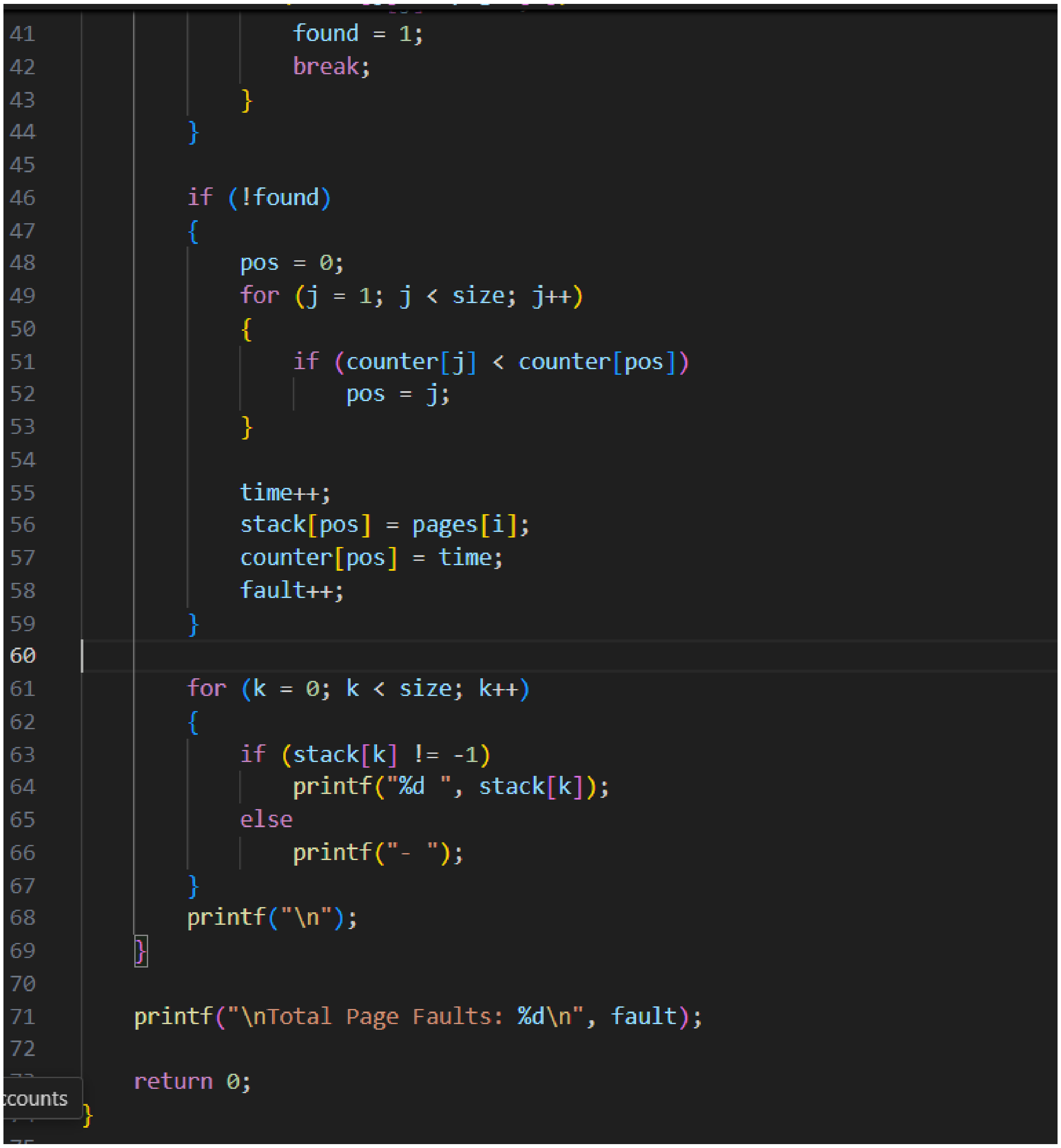
Algorithm:

1. Start the process
2. Declare the size
3. Get the number of pages to be inserted
4. Get the value
5. Declare counter and stack
6. Select the least recently used page by counter value
7. Stack them according to the selection.
8. Display the values
9. Stop the process

Program:

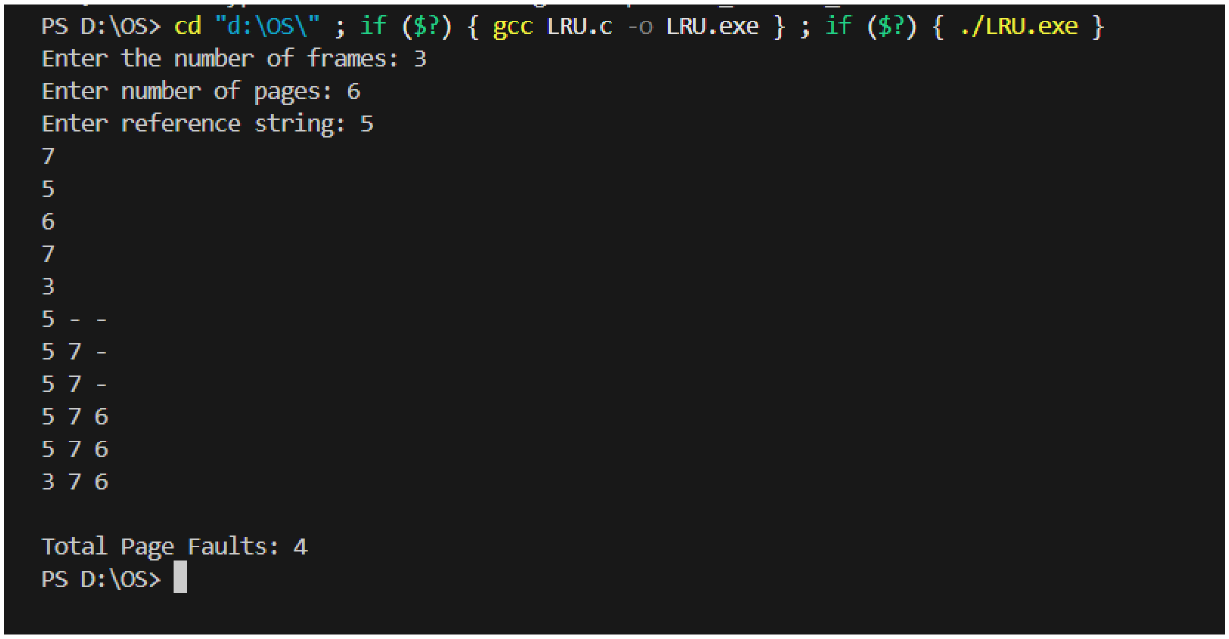


```c
#include <stdio.h>

int main()
{
    int size, n;

    printf("Enter the number of frames: ");
    scanf("%d", &size);

    printf("Enter number of pages: ");
    scanf("%d", &n);

    int pages[n];
    int stack[size];
    int counter[size];

    int i, j, k, pos, fault = 0, time = 0, found;

    printf("Enter reference string: ");
    for (i = 0; i < n; i++)
    {
        scanf("%d", &pages[i]);
    }

    for (i = 0; i < size; i++)
    {
        stack[i] = -1;
        counter[i] = 0;
    }

    for (i = 0; i < n; i++)
    {
        found = 0;

        for (j = 0; j < size; j++)
        {
            if (stack[j] == pages[i])
            {
                time++;
                counter[j] = time;
                found = 1;
```



```
                found = 1;
                break;
            }
        }

        if (!found)
        {
            pos = 0;
            for (j = 1; j < size; j++)
            {
                if (counter[j] < counter[pos])
                    pos = j;
            }

            time++;
            stack[pos] = pages[i];
            counter[pos] = time;
            fault++;
        }

        for (k = 0; k < size; k++)
        {
            if (stack[k] != -1)
                printf("%d ", stack[k]);
            else
                printf("- ");
        }
        printf("\n");
    }

    printf("\nTotal Page Faults: %d\n", fault);

    return 0;
}
```

Output:



```
PS D:\OS> cd "d:\OS\" ; if ($?) { gcc LRU.c -o LRU.exe } ; if ($?) { ./LRU.exe }
Enter the number of frames: 3
Enter number of pages: 6
Enter reference string: 5
7
5
6
7
3
5 - -
5 7 -
5 7 -
5 7 6
5 7 6
3 7 6

Total Page Faults: 4
PS D:\OS>
```

Result:

Hence the number of page faults that occur using LRU page replacement technique has been found successfully.

Ex. No. : 11c **Date : 18.04.2025**

Register No. : 230701385 **Name : VISHWAK S**

OPTIMAL

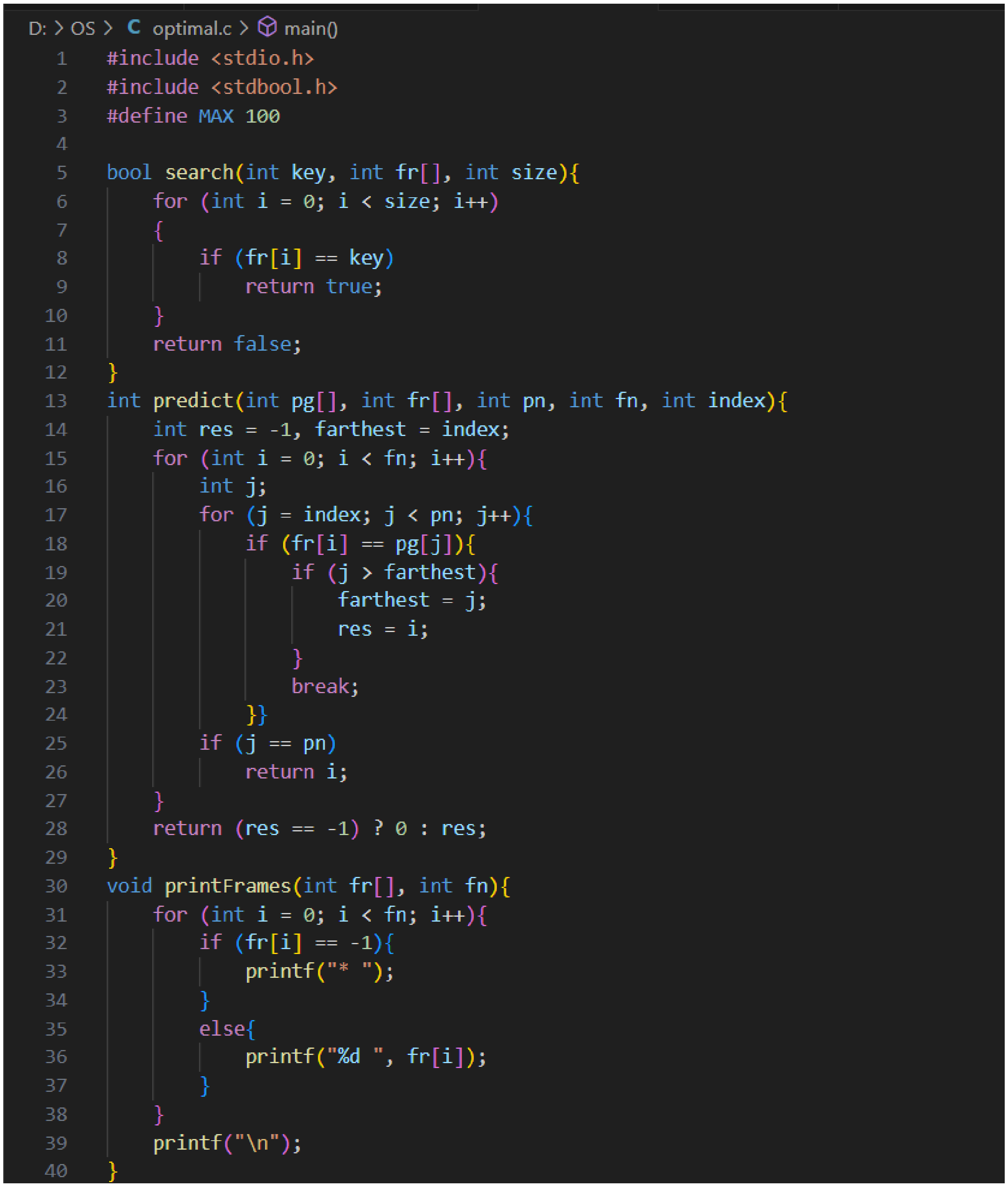
Aim:

To write a c program to implement an Optimal page replacement algorithm.

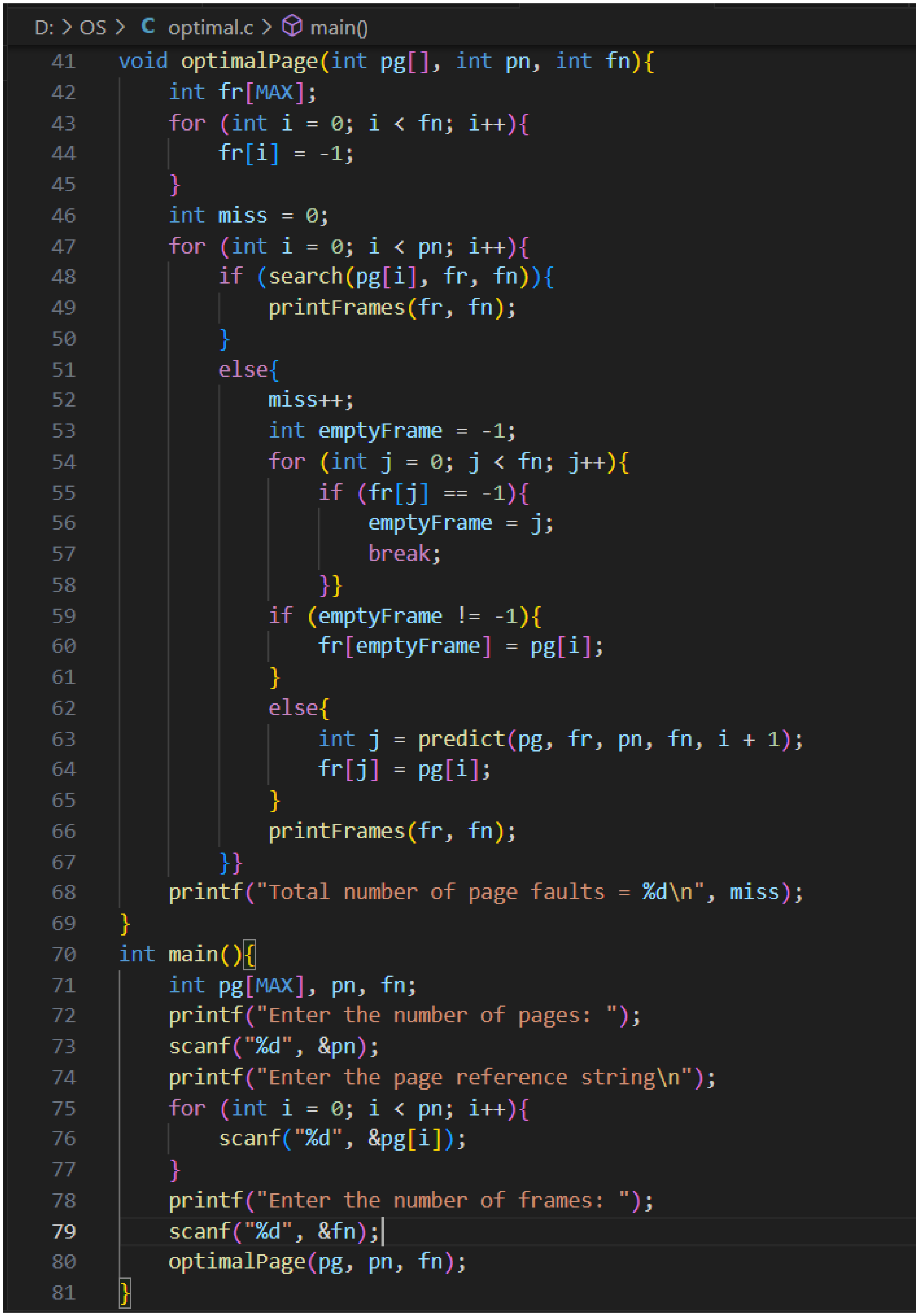
Algorithm:

1. Start the process
2. Declare the size
3. Get the number of pages to be inserted
4. Get the value
5. Declare counter and stack
6. Select the least frequently used page by counter value
7. Stack them according to the selection.
8. Display the values
9. Stop the process

Program:

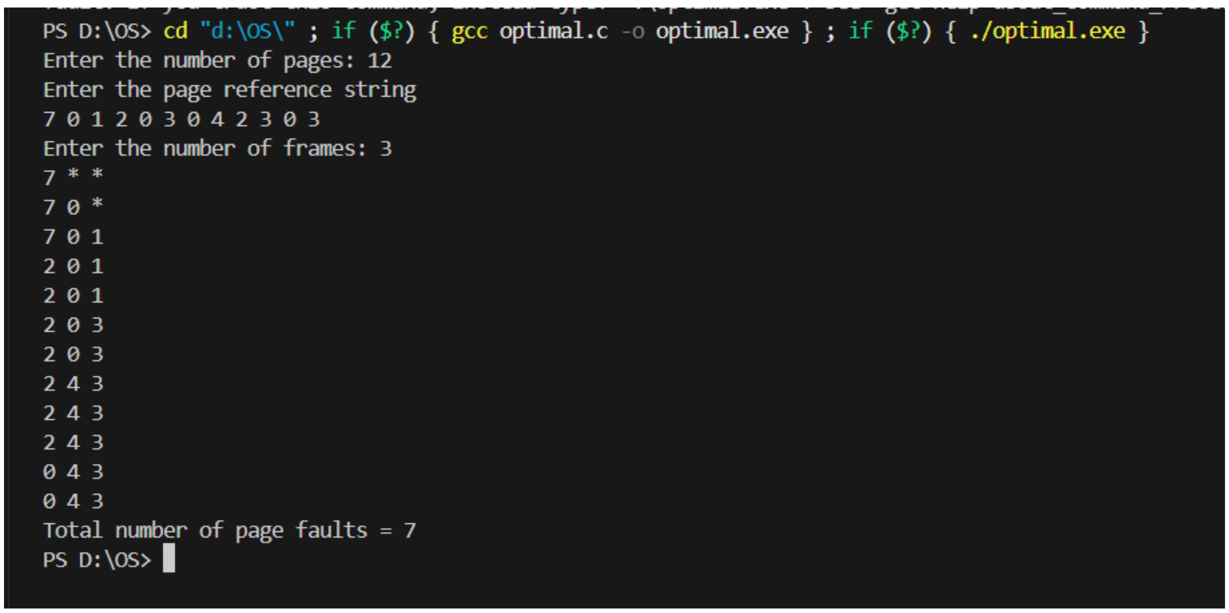

D: > OS > C optimal.c > main()

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

bool search(int key, int fr[], int size){
    for (int i = 0; i < size; i++)
    {
        if (fr[i] == key)
            return true;
    }
    return false;
}
int predict(int pg[], int fr[], int pn, int fn, int index){
    int res = -1, farthest = index;
    for (int i = 0; i < fn; i++){
        int j;
        for (j = index; j < pn; j++){
            if (fr[i] == pg[j]){
                if (j > farthest){
                    farthest = j;
                    res = i;
                }
                break;
            }}
        if (j == pn)
            return i;
    }
    return (res == -1) ? 0 : res;
}
void printFrames(int fr[], int fn){
    for (int i = 0; i < fn; i++){
        if (fr[i] == -1){
            printf("* ");
        }
        else{
            printf("%d ", fr[i]);
        }
    }
    printf("\n");
}
```



```c
void optimalPage(int pg[], int pn, int fn){
    int fr[MAX];
    for (int i = 0; i < fn; i++){
        fr[i] = -1;
    }
    int miss = 0;
    for (int i = 0; i < pn; i++){
        if (search(pg[i], fr, fn)){
            printFrames(fr, fn);
        }
        else{
            miss++;
            int emptyFrame = -1;
            for (int j = 0; j < fn; j++){
                if (fr[j] == -1){
                    emptyFrame = j;
                    break;
                }}
            if (emptyFrame != -1){
                fr[emptyFrame] = pg[i];
            }
            else{
                int j = predict(pg, fr, pn, fn, i + 1);
                fr[j] = pg[i];
            }
            printFrames(fr, fn);
        }}
    printf("Total number of page faults = %d\n", miss);
}
int main(){
    int pg[MAX], pn, fn;
    printf("Enter the number of pages: ");
    scanf("%d", &pn);
    printf("Enter the page reference string\n");
    for (int i = 0; i < pn; i++){
        scanf("%d", &pg[i]);
    }
    printf("Enter the number of frames: ");
    scanf("%d", &fn);
    optimalPage(pg, pn, fn);
}
```

Output:



```
PS D:\OS> cd "d:\OS\" ; if ($?) { gcc optimal.c -o optimal.exe } ; if ($?) { ./optimal.exe }
Enter the number of pages: 12
Enter the page reference string
7 0 1 2 0 3 0 4 2 3 0 3
Enter the number of frames: 3
7 * *
7 0 *
7 0 1
2 0 1
2 0 1
2 0 3
2 0 3
2 4 3
2 4 3
2 4 3
0 4 3
0 4 3
Total number of page faults = 7
PS D:\OS>
```

Result:

Hence the number of page faults that occur using Optimal page replacement technique has been found successfully.