ASSIGNMENT NO: 6

Title: Page Replacement Algorithms

AIM: Implement the C program for Page Replacement Algorithms: FCFS, LRU, and Optimal for frame size as minimum three.

OBJECTIVE:
This assignment helps the students understand the Page Replacement Algorithms in Unix/Linux and how to implement it in C

THEORY:
In an operating system that uses paging for memory management, a page replacement algorithm is needed to decide which page needs to be replaced when a new page comes in.

Page Fault: A page fault happens when a running program accesses a memory page that is mapped into the virtual address space but not loaded in physical memory. Since actual physical memory is much smaller than virtual memory, page faults happen. In case of a page fault, Operating System might have to replace one of the existing pages with the newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce the number of page faults.

Page Replacement Algorithms
There are three types of Page Replacement Algorithms. They are:

- Optimal Page Replacement Algorithm
- First In First Out Page Replacement Algorithm
- Least Recently Used (LRU) Page Replacement Algorithm

<u>First in First out Page Replacement Algorithm</u>
This is the first basic algorithm of Page Replacement Algorithms. This algorithm is basically dependent on the number of frames used. Then each frame takes up the certain page and tries to access it. When the frames are filled then the actual problem starts. The fixed number of frames is filled up with the help of first frames present. This concept is fulfilled with the help of Demand Paging

After filling up of the frames, the next page in the waiting queue tries to enter the frame. If the frame is present then, no problem is occurred. Because of the page which is to be searched is already present in the allocated frames.

If the page to be searched is found among the frames then, this process is known as Page Hit.

If the page to be searched is not found among the frames then, this process is known as Page Fault.

When Page Fault occurs this problem arises, then the First In First Out Page Replacement Algorithm comes into picture.

The First In First Out (FIFO) Page Replacement Algorithm removes the Page in the frame which is allotted long back. This means the useless page which is in the frame for a longer time is removed and the new page which is in the ready queue and is ready to occupy the frame is allowed by the First In First Out Page Replacement.

OPTIMAL Page Replacement Algorithm

This is the second basic algorithm of Page Replacement Algorithms. This algorithm is basically dependent on the number of frames used. Then each frame takes up the certain page and tries to access it. When the frames are filled then the actual problem starts. The fixed number of frames is filled up with the help of first frames present. This concept is fulfilled with the help of Demand Paging

After filling up of the frames, the next page in the waiting queue tries to enter the frame. If the frame is present then, no problem is occurred. Because of the page which is to be searched is already present in the allocated frames.

When Page Fault occurs this problem arises, then the OPTIMAL Page Replacement Algorithm comes into picture.

The OPTIMAL Page Replacement Algorithms works on a certain principle. The principle is:

Replace the Page which is not used in the Longest Dimension of time in future

This principle means that after all the frames are filled then, see the future pages which are to occupy the frames. Go on checking for the pages which are already available in the frames. Choose the page which is at last.

Least Recently Used (LRU) Replacement Algorithm

This is the last basic algorithm of Page Replacement Algorithms. This algorithm is basically dependent on the number of frames used. Then each frame takes up the certain page and tries to access it. When the frames are filled then the actual problem starts. The fixed number of frames is filled up with the help of first frames present. This concept is fulfilled with the help of Demand Paging

After filling up of the frames, the next page in the waiting queue tries to enter the frame. If the frame is present then, no problem is occurred. Because of the page which is to be searched is already present in the allocated frames.

When Page Fault occurs this problem arises, then the Least Recently Used (LRU) Page Replacement Algorithm comes into picture.

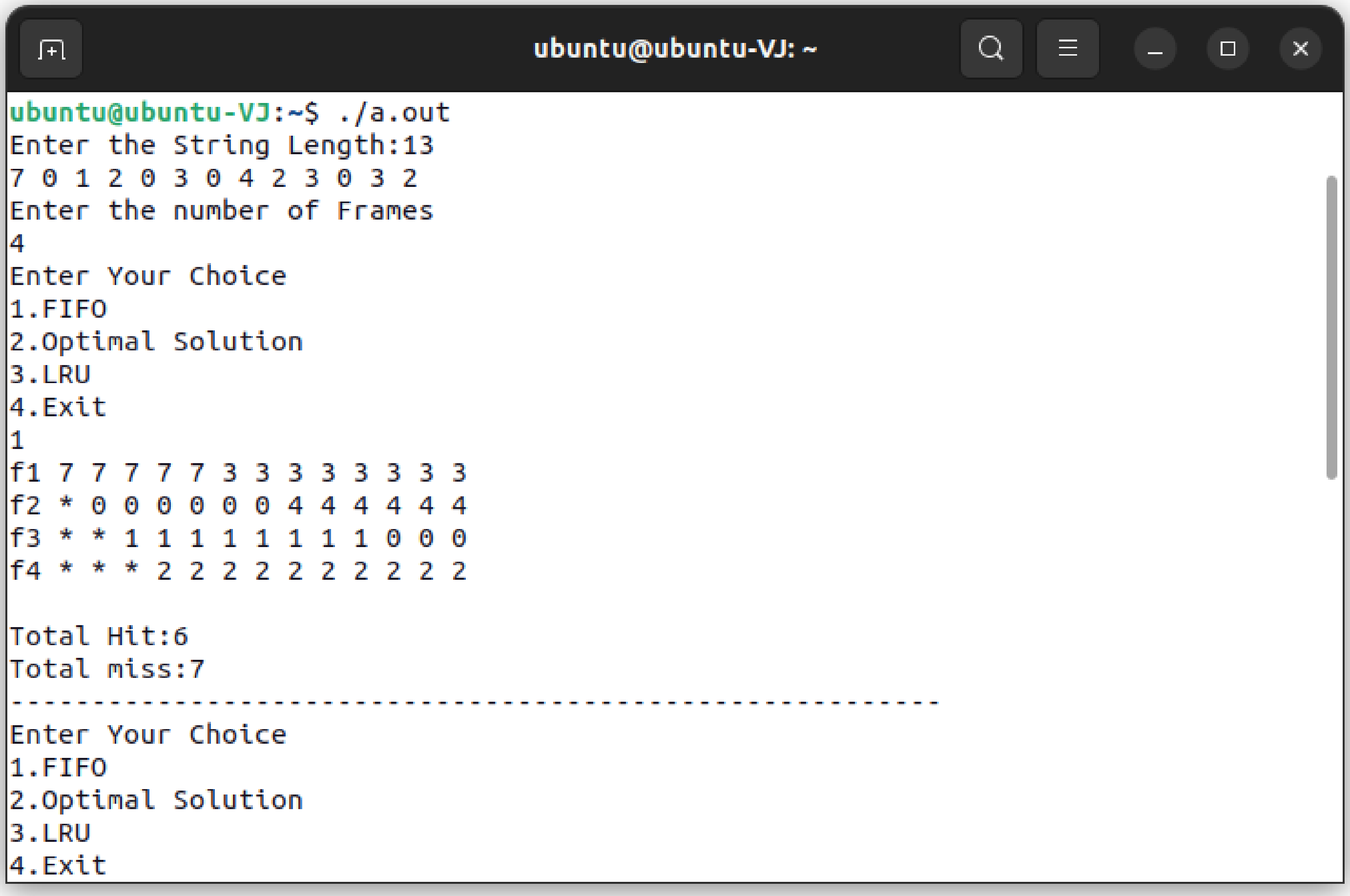
The Least Recently Used (LRU) Page Replacement Algorithms works on a certain principle. The principle is:

Replace the page with the page which is less dimension of time recently used page in the past.

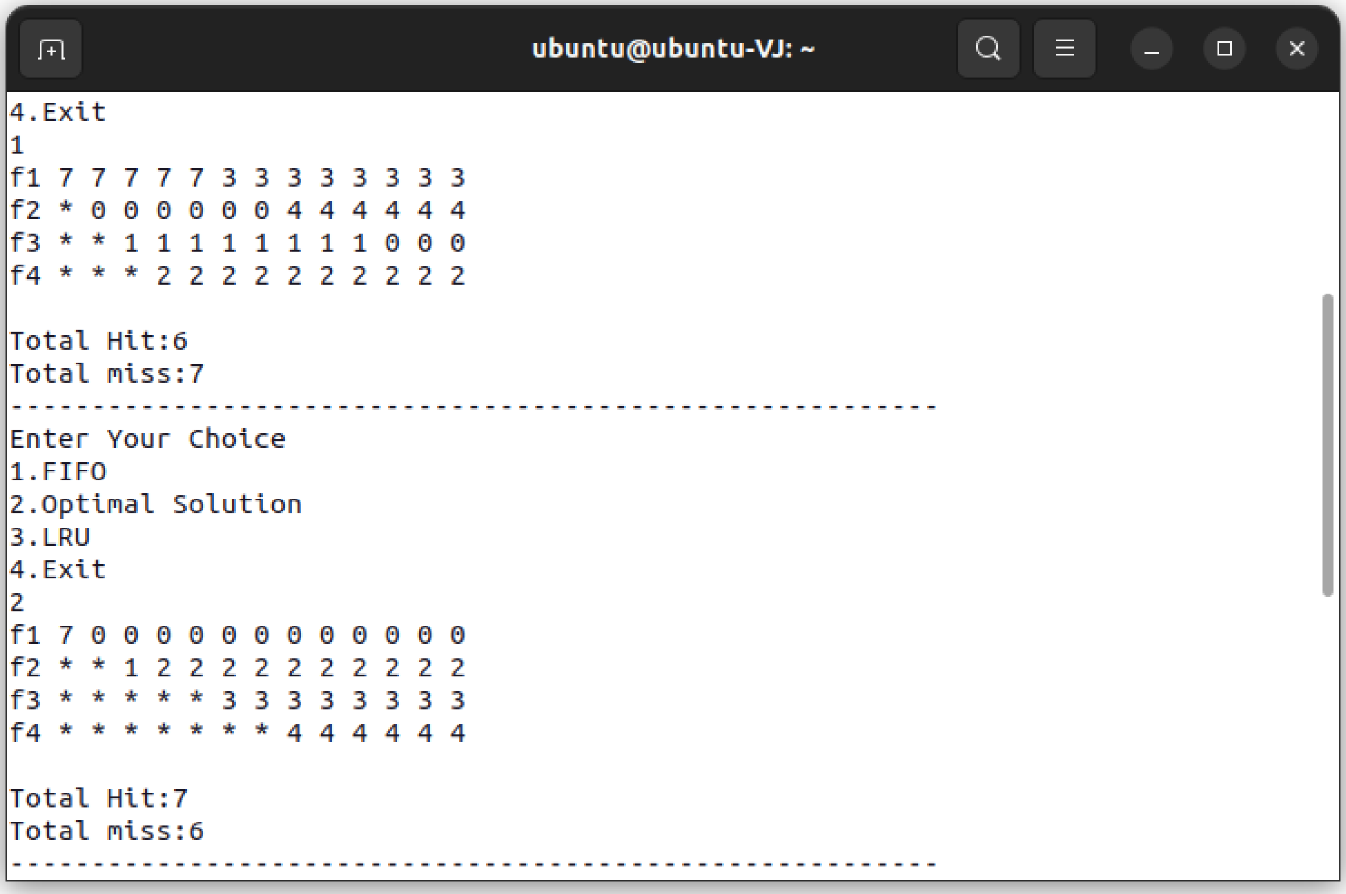
PROGRAM :-

```
#include <stdio.h>

void fifo(int pro[], int q[], int n, int f)
{
    int s = 0; // Initialize the starting index of the queue
    int hit = 0;
    int miss = 0;
    int ans1[f][n]; // Create a 2D array to store the state of frames at each step
    int col = 0;

    // Initialize the queue with -1 values
    for (int i = 0; i < f; i++)
    {
        q[i] = -1;
    }

    for (int i = 0; i < n; i++)
    {
        int cnt = 0;
        // Check if the page is already in the frame
        for (int j = 0; j < f; j++)
        {
            if (pro[i] == q[j])
            {
                cnt++;
                break;
            }
        }
        if (cnt == 0)
        {
            q[s] = pro[i];   // Replace the page at the starting index of the queue
            s = (s + 1) % f; // Update the starting index in a circular manner
            miss++;
        }
        else
        {
            hit++;
        }

        // Record the state of frames at this step
        for (int row = 0; row < f; row++)
        {
            ans1[row][col] = q[row];
        }
        col++;
    }

    // Display the state of frames and the number of hits and misses
    for (int i = 0; i < f; i++)
    {
        printf("f%d ", i + 1);
        for (int j = 0; j < n; j++)
        {
            if (ans1[i][j] == -1)
            {
                printf("* ");
            }
            else
            {
                printf("%d ", ans1[i][j]);
            }
        }
        printf("\n");
    }
    printf("\nTotal Hit:%d", hit);
    printf("\nTotal miss:%d\n");
    printf("-----------------------------------------------------------\n");
}

// The comments for the other two algorithms (OptimalAlgo and LRU) would be similar to the one above, so
I'll omit them here to avoid repetition.

void OptimalAlgo(int pro[], int q[], int n, int f)
{
    // Function code here
}

void LRU(int pro[], int q[], int n, int f)
{
    // Function code here
}

int main()
{
    printf("Enter the String Length:");
    int n;
    scanf("%d", &n);
    int pro[n];
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &pro[i]);
    }
    printf("Enter the number of Frames\n");
    int f;
    scanf("%d", &f);
    int q[f];
    for (int i = 0; i < f; i++)
    {
        q[i] = -1;
    }

    while (1)
    {
        printf("Enter Your Choice\n1.FIFO\n2.Optimal Solution\n3.LRU\n4.Exit\n");
        int var;
        scanf("%d", &var);
        if (var == 4)
        {
            break;
        }

        switch (var)
        {
        case 1:
            fifo(pro, q, n, f);
            break;
        case 2:
            OptimalAlgo(pro, q, n, f);
            break;
        case 3:
            LRU(pro, q, n, f);
            break;
        default:
            printf("Wrong Response");
            break;
        }
    }

    return 0;
}
```

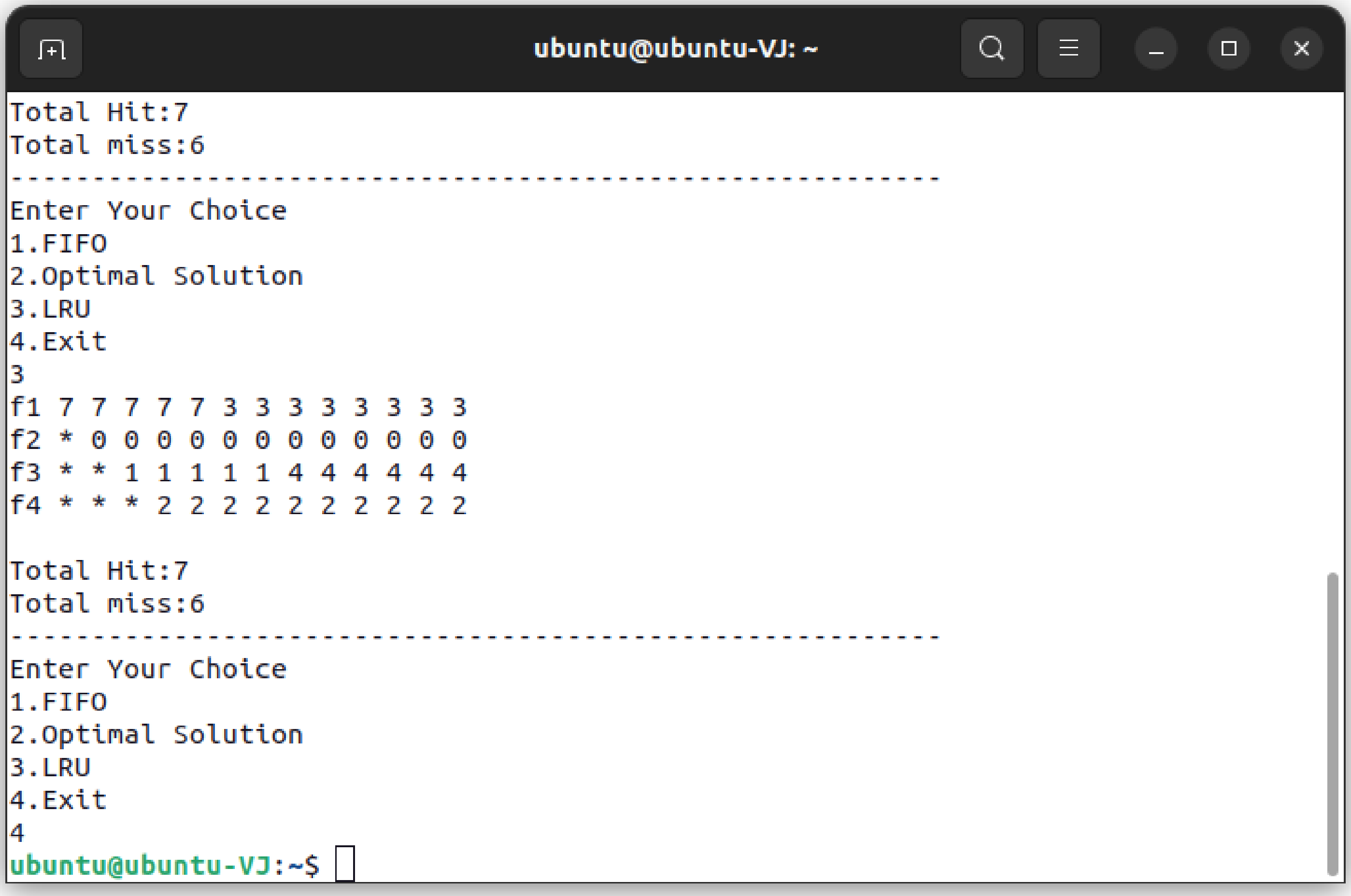
OUTPUT:-



```
ubuntu@ubuntu-VJ:~$ ./a.out
Enter the String Length:13
7 0 1 2 0 3 0 4 2 3 0 3 2
Enter the number of Frames
4
Enter Your Choice
1.FIFO
2.Optimal Solution
3.LRU
4.Exit
1
f1 7 7 7 7 7 3 3 3 3 3 3 3 3
f2 * 0 0 0 0 0 0 4 4 4 4 4 4
f3 * * 1 1 1 1 1 1 1 1 0 0 0
f4 * * * 2 2 2 2 2 2 2 2 2 2

Total Hit:6
Total miss:7
---------------------------------------------------------
Enter Your Choice
1.FIFO
2.Optimal Solution
3.LRU
4.Exit
```



```
4.Exit
1
f1 7 7 7 7 7 3 3 3 3 3 3 3 3
f2 * 0 0 0 0 0 0 4 4 4 4 4 4
f3 * * 1 1 1 1 1 1 1 1 0 0 0
f4 * * * 2 2 2 2 2 2 2 2 2 2

Total Hit:6
Total miss:7
-----------------------------------------------------------
Enter Your Choice
1.FIFO
2.Optimal Solution
3.LRU
4.Exit
2
f1 7 0 0 0 0 0 0 0 0 0 0 0 0
f2 * * 1 2 2 2 2 2 2 2 2 2 2
f3 * * * * * 3 3 3 3 3 3 3 3
f4 * * * * * * * 4 4 4 4 4 4

Total Hit:7
Total miss:6
-----------------------------------------------------------
```



```
Total Hit:7
Total miss:6
-----------------------------------------------------------
Enter Your Choice
1.FIFO
2.Optimal Solution
3.LRU
4.Exit
3
f1 7 7 7 7 7 3 3 3 3 3 3 3 3
f2 * 0 0 0 0 0 0 0 0 0 0 0 0
f3 * * 1 1 1 1 1 4 4 4 4 4 4
f4 * * * 2 2 2 2 2 2 2 2 2 2

Total Hit:7
Total miss:6
-----------------------------------------------------------
Enter Your Choice
1.FIFO
2.Optimal Solution
3.LRU
4.Exit
4
ubuntu@ubuntu-VJ:~$
```