Assignment No. 6

Name: **Sahil Jagadale** Division: TE-11 Batch: L-11 Roll No.: **33327**

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:

Page Replacement technique uses the following approach. If there is no free frame, then we will find the one that is not currently being used and then free it. A-frame can be freed by writing its content to swap space and then change the page table in order to indicate that the page is no longer in the memory.

1. First of all, find the location of the desired page on the disk.

2. Find a free Frame:

a) If there is a free frame, then use it.

b) If there is no free frame then make use of the page-replacement algorithm in order to select the victim frame.

c) Then after that write the victim frame to the disk and then make the changes in the page table and frame table accordingly.

3. After that read the desired page into the newly freed frame and then change the page and frame tables.

4. Restart the process.

1. FIFO Page Replacement Algorithm

It is a very simple way of Page replacement and is referred to as First in First Out. This algorithm mainly replaces the oldest page that has been present in the main memory for the longest time. This algorithm is implemented by keeping the track of all the pages in the queue. As new pages are requested and are swapped in, they are added to the tail of a queue and the page which is at the head becomes the victim. This is not an effective way of page replacement but it can be used for small systems.

2. LRU Page Replacement Algorithm

This algorithm stands for "Least recent used" and this algorithm helps the Operating system to search those pages that are used over a short duration of time frame.

- The page that has not been used for the longest time in the main memory will be selected for replacement.
- This algorithm is easy to implement.
- This algorithm makes use of the counter along with the even-page.

3. Optimal Page Replacement Algorithm

This algorithm mainly replaces the page that will not be used for the longest time in the future. The practical implementation of this algorithm is not possible.

• Practical implementation is not possible because we cannot predict in advance those pages that will not be used for the longest time in the future.

• This algorithm leads to less number of page faults and thus is the best-known algorithm

Also, this algorithm can be used to measure the performance of other algorithms.

Code:

```
#include <stdio.h>

void fifo(int pro[], int q[], int n, int f)
{
         int s = 0;
         int hit = 0;
         int miss = 0;
         int ans1[f][n];
         int col = 0;
         for (int i = 0; i < f; i++){
                q[i] = -1;
         }
         for (int i = 0; i < n; i++){
                 int cnt = 0;
                 for (int j = 0; j < f; j++){
                         if (pro[i] == q[j]){
                                 cnt++;
                                 break;
                         }
                 }
                 if (cnt == 0){
                         q[s] = pro[i];
                         s = (s + 1) % f;
                         miss++;
                 }
                 else{
                        hit++;
                 }
                 for (int row = 0; row < f; row++){
                        ans1[row][col] = q[row];
                 }
                 col++;
         }

         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", miss);
         printf("-----------------------------------------------------------\n");
}
void OptimalAlgo(int pro[], int q[], int n, int f)
{
         int hit = 0;
```

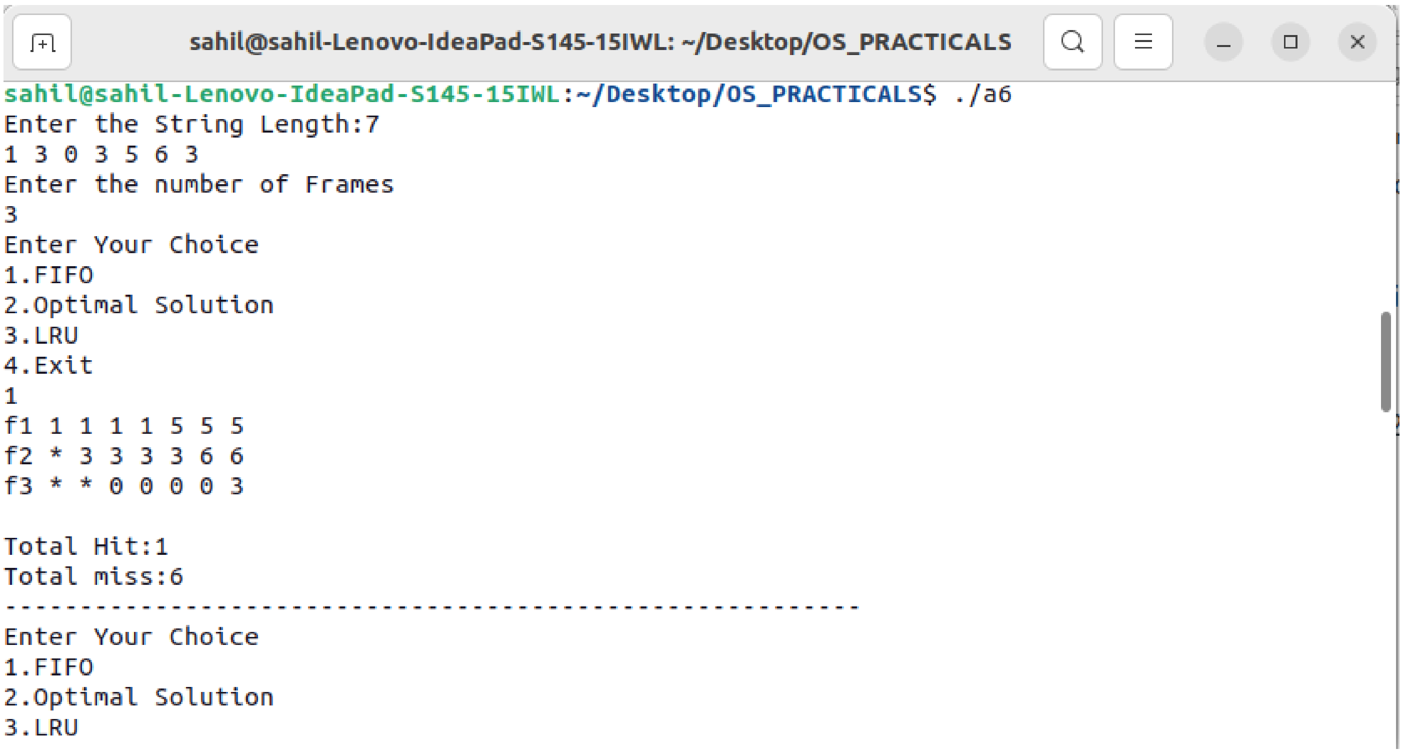
```
        int miss = 0;
        int ans2[f][n];
        int col = 0;
        for (int i = 0; i < f; i++){
               q[i] = -1;
        }
        for (int i = 0; i < n; i++){
                int cnt = 0;
                for (int j = 0; j < f; j++){
                        if (pro[i] == q[j]){
                                cnt++;
                                break;
                        }
                }
                if (cnt != 0){
                       hit++;
                }
                else{
                        miss++;
                        int v = -1;
                        int ok = 0;
                        int s[f]; // Create an array to simulate a set
                        for (int k = 0; k < f; k++){
                               s[k] = -1; // Initialize the set with -1 values
                        }
                        for (int j = i + 1; j < n; j++){
                                for (int k = 0; k < f; k++){
                                        if (pro[j] == q[k]){
                                                v = pro[j];
                                                s[k] = v; // Add the element to the set
                                                break;
                                        }
                                }
                                int set_full = 1;
                                for (int k = 0; k < f; k++){
                                        if (s[k] == -1){
                                                set_full = 0; // The set is not full yet
                                                break;
                                        }
                                }
                                if (set_full){
                                        ok = 1;
                                        break;
                                }
                       }
                       if (ok){
                               if (v != -1){
                                        int x = -1;
                                        for (int j = 0; j < f; j++){
                                                if (q[j] == -1){
                                                        x = j;
                                                        break;
                                                }
                                        }
                                        if (x != -1){
                                               q[x] = pro[i];
                                        }
                                        else{
                                                for (int j = 0; j < f; j++){
                                                        if (q[j] == v){
                                                                q[j] = pro[i];
                                                                break;
                                                        }
                                                }
                                        }
                                }
                                else{
```

```c
                                        int x = -1;
                                        for (int j = 0; j < f; j++){
                                                if (q[j] == -1){
                                                        x = j;
                                                        break;
                                                }
                                        }
                                        if (x != -1){
                                                q[x] = pro[i];
                                        }
                                        else{
                                                q[0] = pro[i];
                                        }
                                }
                        }
                        else{
                                for (int j = 0; j < f; j++){
                                        if (s[j] == -1){
                                                q[j] = pro[i];
                                                break;
                                        }
                                }
                        }
                }
                for (int row = 0; row < f; row++){
                        ans2[row][col] = q[row];
                }
                col++;
        }
        for (int i = 0; i < f; i++){
                printf("f%d ", i + 1);
                for (int j = 0; j < n; j++){
                        if (ans2[i][j] == -1){
                                printf("* ");
                        }
                        else{
                                printf("%d ", ans2[i][j]);
                        }
                }
                printf("\n");
        }
        printf("\nTotal Hit:%d", hit);
        printf("\nTotal miss:%d\n", miss);
        printf("------------------------------------------------------------\n");
}

void LRU(int pro[], int q[], int n, int f)
{
        int hit = 0;
        int miss = 0;
        int ans3[f][n];
        for (int i = 0; i < f; i++){
                q[i] = -1;
        }
        int col = 0;
        for (int i = 0; i < n; i++){
                int cnt = 0;
                for (int j = 0; j < f; j++){
                        if (pro[i] == q[j]){
                                cnt++;
                                break;
                        }
                }
                if (cnt != 0){
                        hit++;
                }
                else{
```

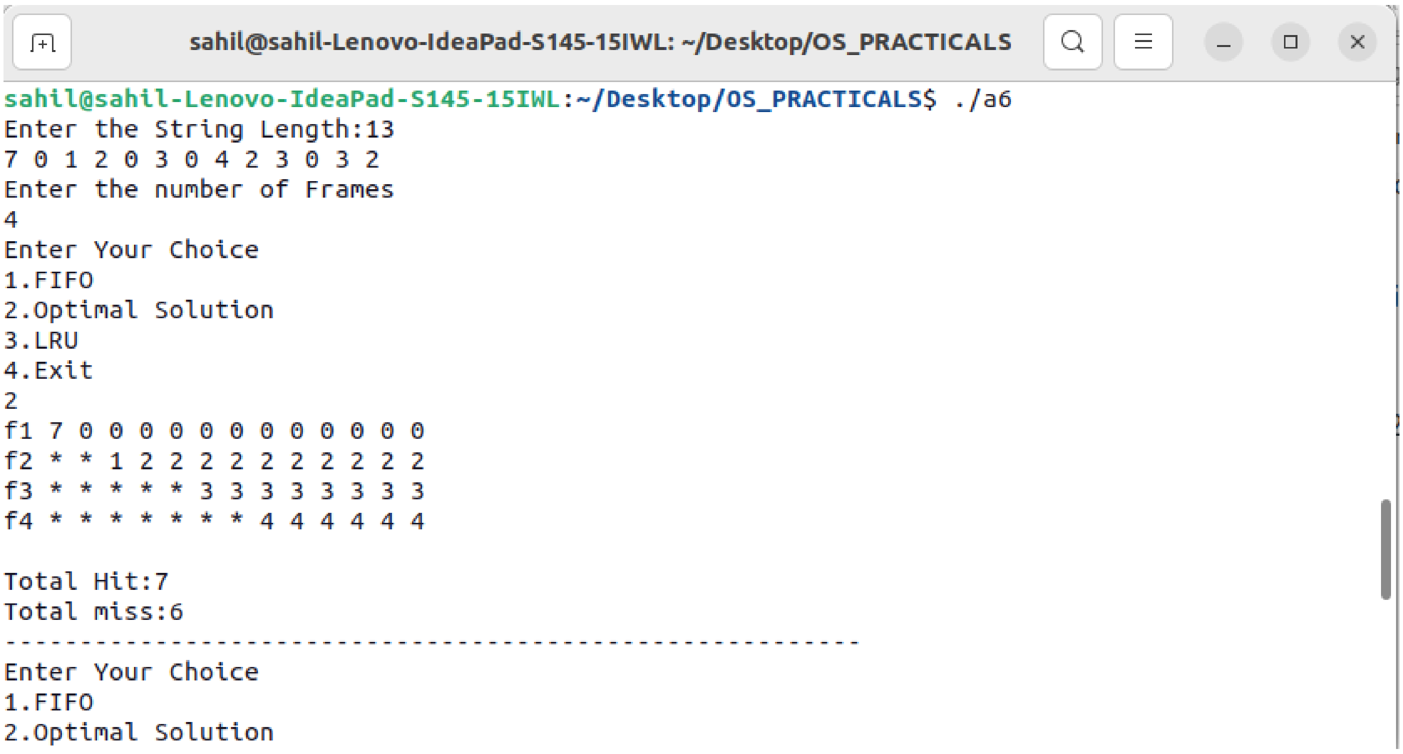
```
                miss++;
                int v = -1;
                int s[f]; // Create an array to simulate a set
                for (int k = 0; k < f; k++){
                    s[k] = -1; // Initialize the set with -1 values
                }
                for (int j = i - 1; j >= 0 && i != 0; j--){
                        for (int k = 0; k < f; k++){
                                if (pro[j] == q[k]){
                                        v = pro[j];
                                        s[k] = v; // Add the element to the set
                                        break;
                                }
                        }
                        int set_full = 1;
                        for (int k = 0; k < f; k++){
                                if (s[k] == -1){
                                        set_full = 0; // The set is not full yet
                                        break;
                                }
                        }
                        if (set_full){
                            break;
                        }
                }
                if (v != -1){
                        int x = -1;
                        for (int j = 0; j < f; j++){
                                if (q[j] == -1){
                                        x = j;
                                        break;
                                }
                        }
                        if (x != -1){
                            q[x] = pro[i];
                        }
                        else{
                                for (int j = 0; j < f; j++){
                                        if (q[j] == v){
                                                q[j] = pro[i];
                                                break;
                                        }
                                }
                        }
                }
                else{
                        int x = -1;
                        for (int j = 0; j < f; j++){
                                if (q[j] == -1){
                                        x = j;
                                        break;
                                }
                        }
                        if (x != -1){
                            q[x] = pro[i];
                        }
                        else{
                            q[0] = pro[i];
                        }
                }
        }
        for (int row = 0; row < f; row++){
            ans3[row][col] = q[row];
        }
        col++;
    }
    for (int i = 0; i < f; i++){
```

```
                printf("f%d ", i + 1);
                for (int j = 0; j < n; j++){
                        if (ans3[i][j] == -1){
                                printf("* ");
                        }
                        else{
                                printf("%d ", ans3[i][j]);
                        }
                }
                printf("\n");
        }
        printf("\nTotal Hit:%d", hit);
        printf("\nTotal miss:%d\n", miss);
        printf("------------------------------------------------------------\n");
}
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;
}
```

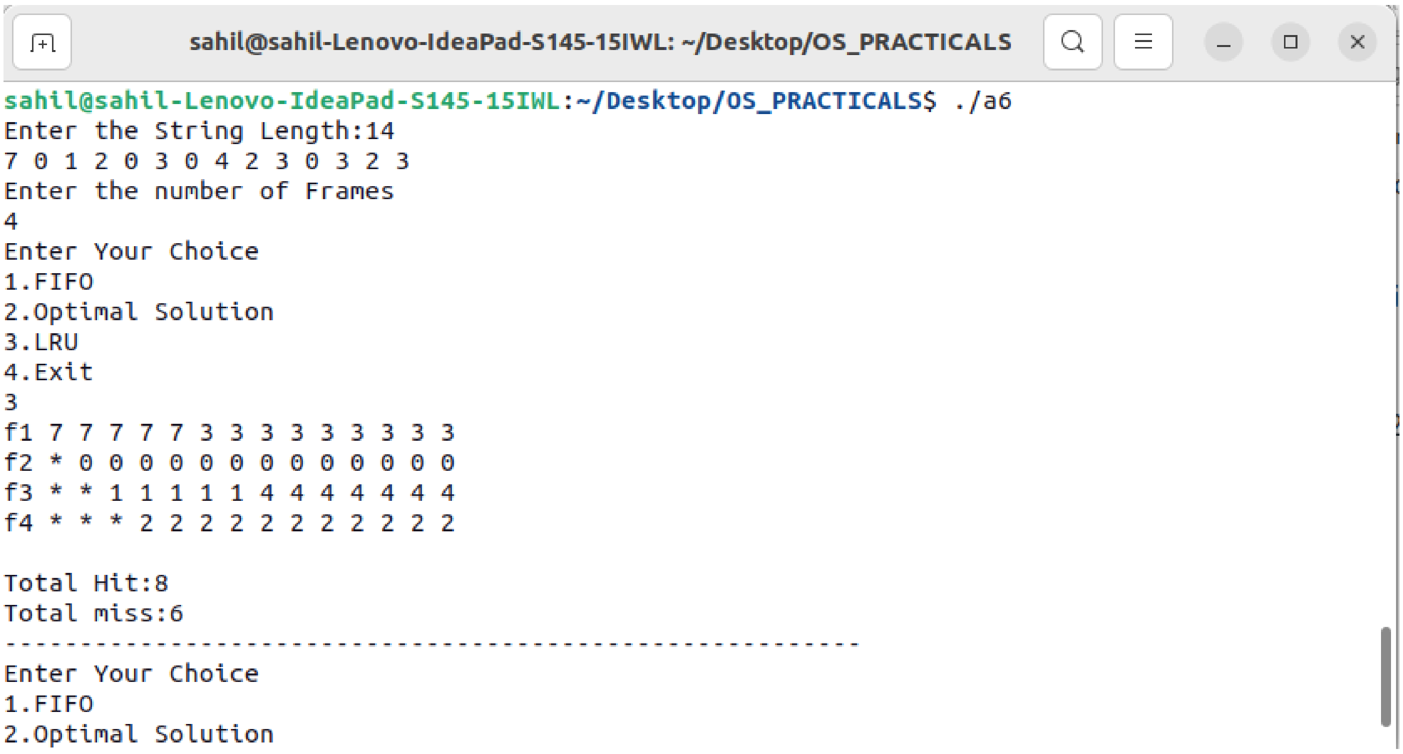
Output:


sahil@sahil-Lenovo-IdeaPad-S145-15IWL: ~/Desktop/OS_PRACTICALS

```
sahil@sahil-Lenovo-IdeaPad-S145-15IWL:~/Desktop/OS_PRACTICALS$ ./a6
Enter the String Length:7
1 3 0 3 5 6 3
Enter the number of Frames
3
Enter Your Choice
1.FIFO
2.Optimal Solution
3.LRU
4.Exit
1
f1 1 1 1 1 5 5 5
f2 * 3 3 3 3 6 6
f3 * * 0 0 0 0 3

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


sahil@sahil-Lenovo-IdeaPad-S145-15IWL: ~/Desktop/OS_PRACTICALS

```
sahil@sahil-Lenovo-IdeaPad-S145-15IWL:~/Desktop/OS_PRACTICALS$ ./a6
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
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
---------------------------------------------------------
Enter Your Choice
1.FIFO
2.Optimal Solution
```


sahil@sahil-Lenovo-IdeaPad-S145-15IWL: ~/Desktop/OS_PRACTICALS

```
sahil@sahil-Lenovo-IdeaPad-S145-15IWL:~/Desktop/OS_PRACTICALS$ ./a6
Enter the String Length:14
7 0 1 2 0 3 0 4 2 3 0 3 2 3
Enter the number of Frames
4
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 3
f2 * 0 0 0 0 0 0 0 0 0 0 0 0 0
f3 * * 1 1 1 1 1 4 4 4 4 4 4 4
f4 * * * 2 2 2 2 2 2 2 2 2 2 2

Total Hit:8
Total miss:6
---------------------------------------------------------
Enter Your Choice
1.FIFO
2.Optimal Solution
```