ASSIGNMENT - IV

TITLE :

Implement a simulation of virtual memory using demand paging, and apply page replacement algorithms such as FIFO (First-In-First-Out),LRU (Least Recently Used) and optimal to handle page faults and minimize them.

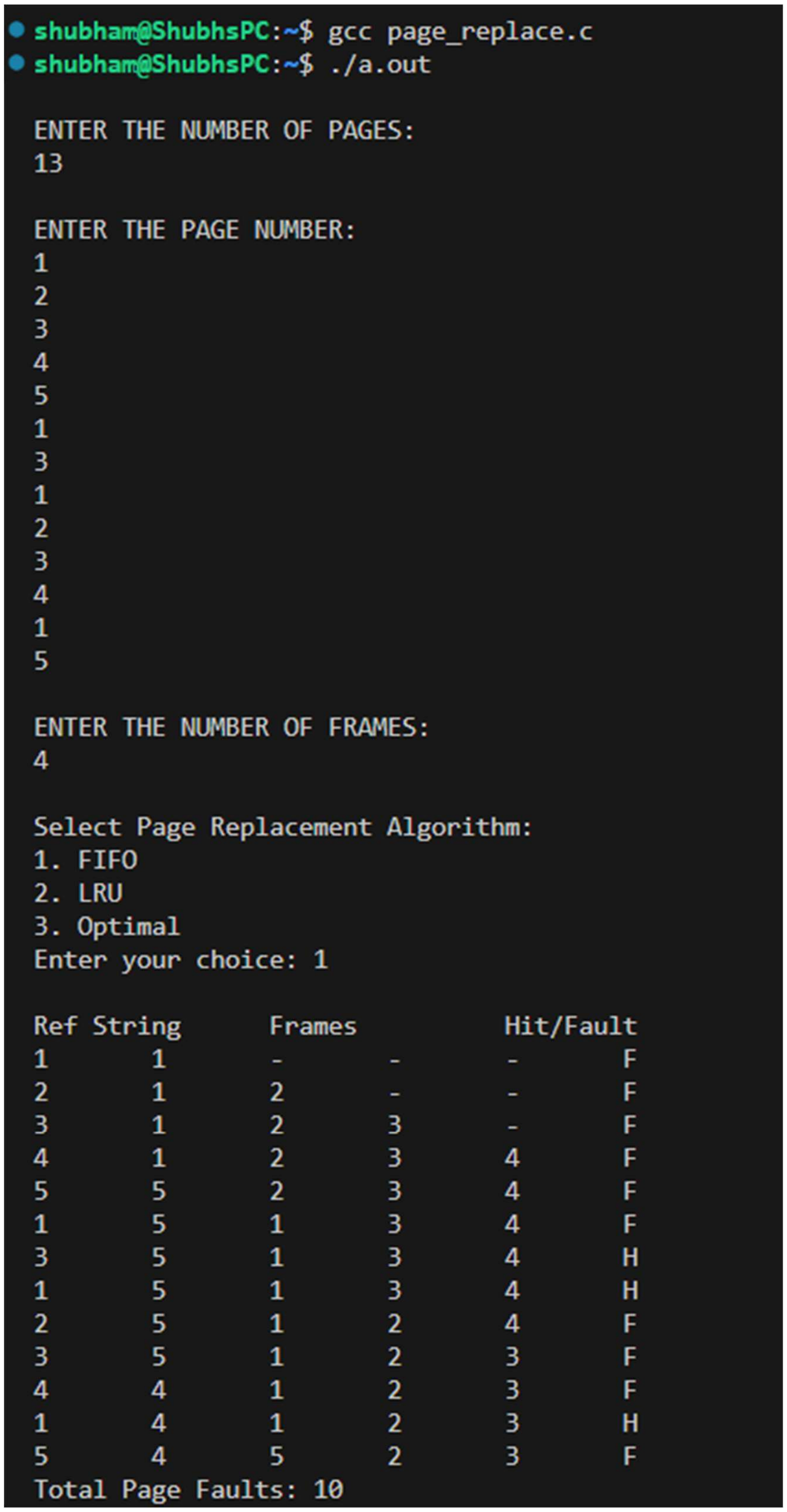
NAME : Shinde Shubham Dnyandev, **DIV :** SY-B, **ROLL NO. :** 23107121.

PROGRAM :

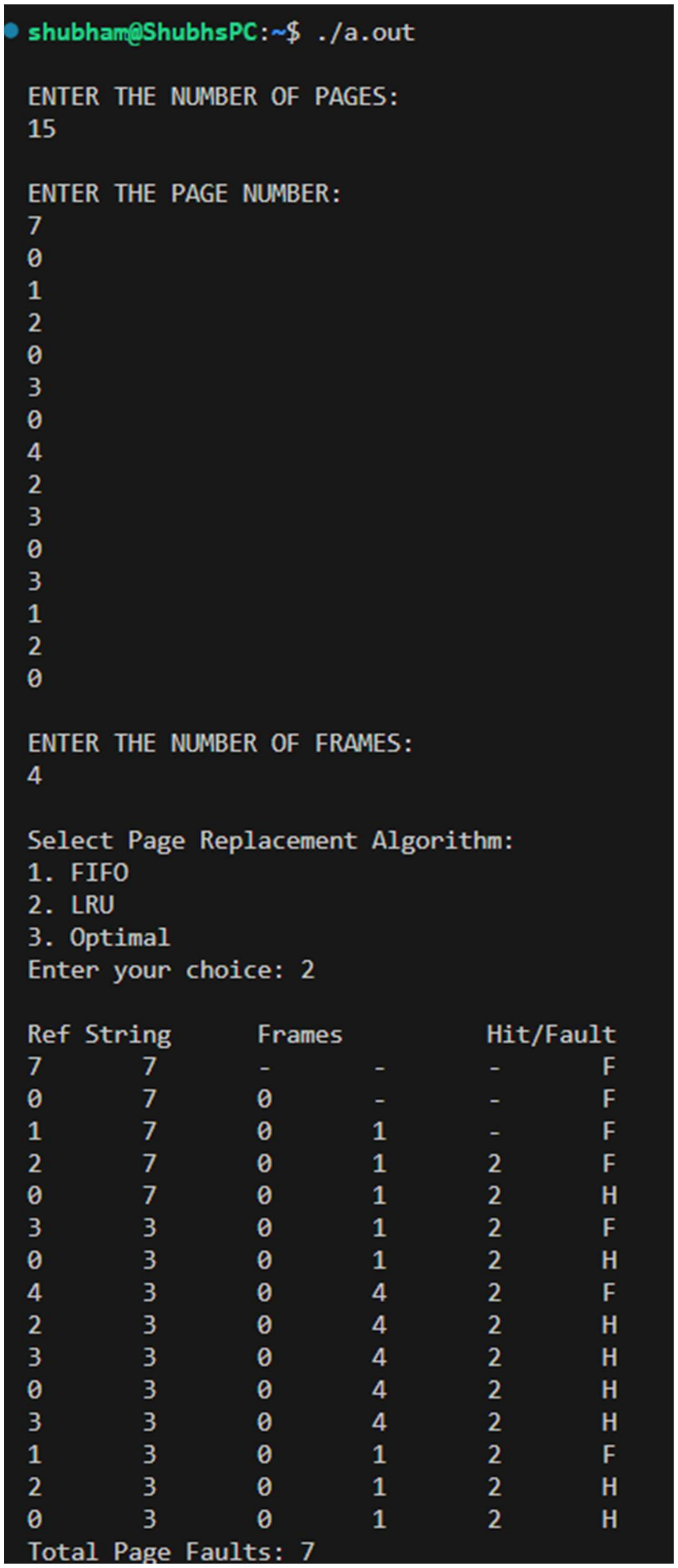
```
#include <stdio.h>

void fifo(int ref_str[], int n, int no);
void lru(int ref_str[], int n, int no);
void optimal(int ref_str[], int n, int no);
int pageinframe(int frame[], int no, int page);
void printFrames(int frame[], int no, int flag);
int findfuture(int ref_str[], int frame[], int n, int currentIndex, int no);

int main() {
    int n, no, choice;
    int ref_str[50];

    printf("\nENTER THE NUMBER OF PAGES:\n");
    scanf("%d", &n);

    printf("\nENTER THE PAGE NUMBER:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &ref_str[i]);
    }

    printf("\nENTER THE NUMBER OF FRAMES:\n");
    scanf("%d", &no);

    printf("\nSelect Page Replacement Algorithm:\n");
    printf("1. FIFO\n2. LRU\n3. Optimal\nEnter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            fifo(ref_str, n, no);
            break;
        case 2:
            lru(ref_str, n, no);
            break;
        case 3:
            optimal(ref_str, n, no);
            break;
        default:
            printf("Invalid choice! Exiting.\n");
    }

    return 0;
}

int pageinframe(int frame[], int no, int page) {
    for (int i = 0; i < no; i++) {
        if (frame[i] == page) {
            return 1;
        }
    }
    return 0;
}

void printFrames(int frame[], int no, int flag) {
    for (int i = 0; i < no; i++) {
        if (frame[i] == -1) {
            printf("-\t");
        } else {
            printf("%d\t", frame[i]);
        }
    }

    if (flag == 1) {
        printf("H\n");
    } else {
        printf("F\n");
    }
}

void fifo(int ref_str[], int n, int no) {
    int frame[10], j = 0, fcount = 0;
    for (int i = 0; i < no; i++) {
        frame[i] = -1;
    }

    printf("\nRef String\tFrames\t\tHit/Fault\n");

    for (int i = 0; i < n; i++) {
        printf("%d\t", ref_str[i]);

        int flag = pageinframe(frame, no, ref_str[i]);

        if (flag == 0) {
            frame[j] = ref_str[i];
            j = (j + 1) % no;
            fcount++;
        }

        printFrames(frame, no, flag);
    }
    printf("Total Page Faults: %d\n", fcount);
}

void lru(int ref_str[], int n, int no) {
    int frame[10], count[10], fcount = 0, next = 1;

    for (int i = 0; i < no; i++) {
        frame[i] = -1;
        count[i] = 0;
    }

    printf("\nRef String\tFrames\t\tHit/Fault\n");

    for (int i = 0; i < n; i++) {
        printf("%d\t", ref_str[i]);

        int flag = pageinframe(frame, no, ref_str[i]);

        if (flag == 1) {
            for (int j = 0; j < no; j++) {
                if (frame[j] == ref_str[i]) {
                    count[j] = next++;
                    break;
                }
            }
        } else {
            int min = 0;
            for (int j = 1; j < no; j++) {
                if (count[j] < count[min]) {
                    min = j;
                }
            }
            frame[min] = ref_str[i];
            count[min] = next++;
            fcount++;
        }

        printFrames(frame, no, flag);
    }

    printf("Total Page Faults: %d\n", fcount);
}

int findfuture(int ref_str[], int frame[], int n, int currentIndex, int no) {
    int temp = -1, replaceIndex = -1;

    for (int i = 0; i < no; i++) {
        int j;
        for (j = currentIndex + 1; j < n; j++) {
            if (frame[i] == ref_str[j]) {
                if (j > temp) {
                    temp = j;
                    replaceIndex = i;
                }
                break;
            }
        }
        if (j == n) {
            return i;
        }
    }

    if (replaceIndex == -1) {
        return 0;
    }
    return replaceIndex;
```

```
}

void optimal(int ref_str[], int n, int no) {
    int frame[10], fcount = 0;

    for (int i = 0; i < no; i++) {
        frame[i] = -1;
    }

    printf("\nRef String\tFrames\t\tHit/Fault\n");

    for (int i = 0; i < n; i++) {
        printf("%d\t", ref_str[i]);

        int flag = pageinframe(frame, no, ref_str[i]);

        if (flag == 0) {
            if (i < no) {
                frame[i] = ref_str[i];
            } else {
                int replaceIndex = findfuture(ref_str, frame, n, i, no);
                frame[replaceIndex] = ref_str[i];
            }
            fcount++;
        }

        printFrames(frame, no, flag);
    }

    printf("Total Page Faults: %d\n", fcount);
}
```

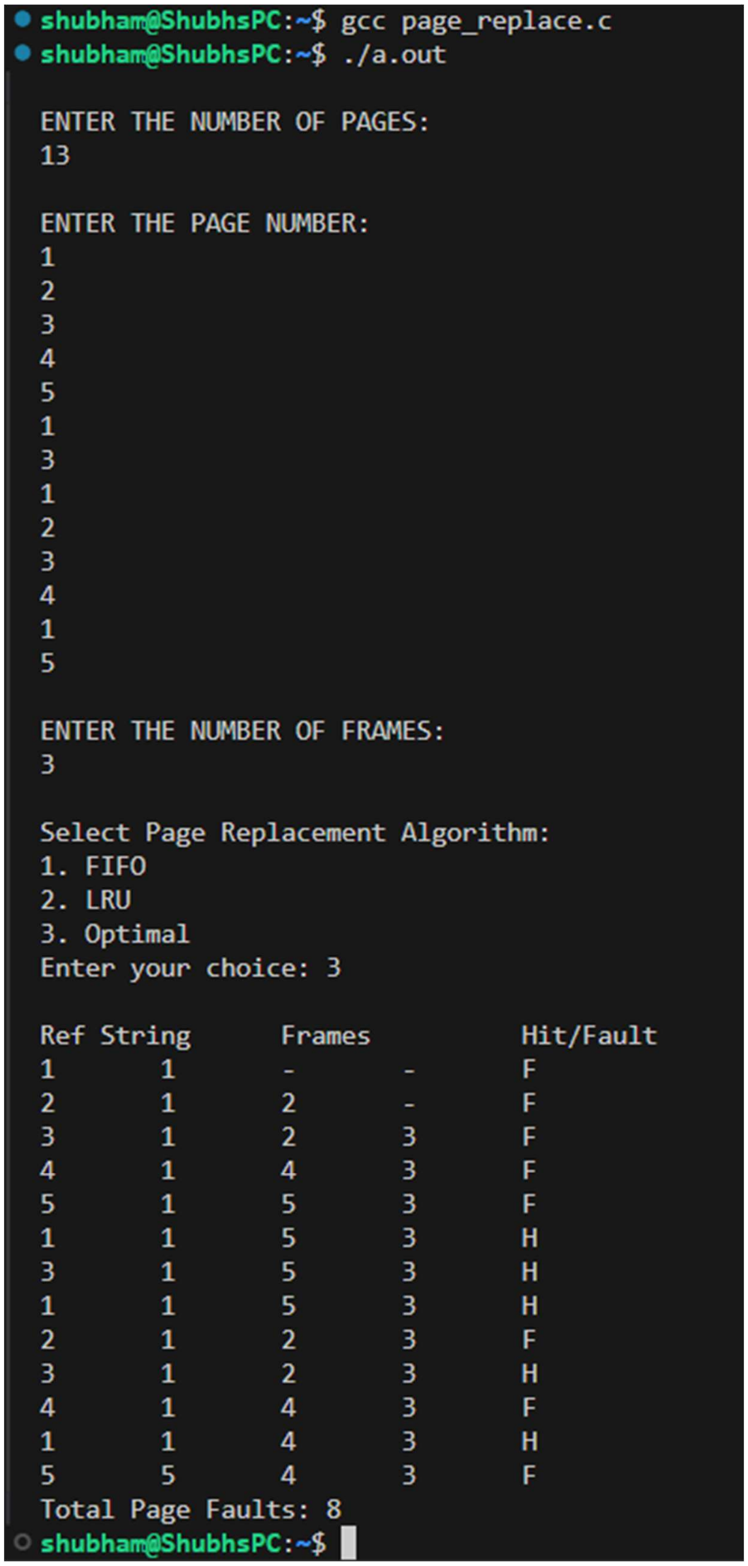
OUTPUT :



```
● shubham@ShubhsPC:~$ gcc page_replace.c
● shubham@ShubhsPC:~$ ./a.out

 ENTER THE NUMBER OF PAGES:
 13

 ENTER THE PAGE NUMBER:
 1
 2
 3
 4
 5
 1
 3
 1
 2
 3
 4
 1
 5

 ENTER THE NUMBER OF FRAMES:
 4

 Select Page Replacement Algorithm:
 1. FIFO
 2. LRU
 3. Optimal
 Enter your choice: 1

 Ref String      Frames          Hit/Fault
 1       1       -       -       -       F
 2       1       2       -       -       F
 3       1       2       3       -       F
 4       1       2       3       4       F
 5       5       2       3       4       F
 1       5       1       3       4       F
 3       5       1       3       4       H
 1       5       1       3       4       H
 2       5       1       2       4       F
 3       5       1       2       3       F
 4       4       1       2       3       F
 1       4       1       2       3       H
 5       4       5       2       3       F
 Total Page Faults: 10
```



```
shubham@ShubhsPC:~$ ./a.out

ENTER THE NUMBER OF PAGES:
15

ENTER THE PAGE NUMBER:
7
0
1
2
0
3
0
4
2
3
0
3
1
2
0

ENTER THE NUMBER OF FRAMES:
4

Select Page Replacement Algorithm:
1. FIFO
2. LRU
3. Optimal
Enter your choice: 2

Ref String      Frames          Hit/Fault
7       7       -       -       -       F
0       7       0       -       -       F
1       7       0       1       -       F
2       7       0       1       2       F
0       7       0       1       2       H
3       3       0       1       2       F
0       3       0       1       2       H
4       3       0       4       2       F
2       3       0       4       2       H
3       3       0       4       2       H
0       3       0       4       2       H
3       3       0       4       2       H
1       3       0       1       2       F
2       3       0       1       2       H
0       3       0       1       2       H
Total Page Faults: 7
```



```
● shubham@ShubhsPC:~$ gcc page_replace.c
● shubham@ShubhsPC:~$ ./a.out

  ENTER THE NUMBER OF PAGES:
  13

  ENTER THE PAGE NUMBER:
  1
  2
  3
  4
  5
  1
  3
  1
  2
  3
  4
  1
  5

  ENTER THE NUMBER OF FRAMES:
  3

  Select Page Replacement Algorithm:
  1. FIFO
  2. LRU
  3. Optimal
  Enter your choice: 3

  Ref String      Frames          Hit/Fault
  1       1       -       -       F
  2       1       2       -       F
  3       1       2       3       F
  4       1       4       3       F
  5       1       5       3       F
  1       1       5       3       H
  3       1       5       3       H
  1       1       5       3       H
  2       1       2       3       F
  3       1       2       3       H
  4       1       4       3       F
  1       1       4       3       H
  5       5       4       3       F
  Total Page Faults: 8
○ shubham@ShubhsPC:~$ 
```