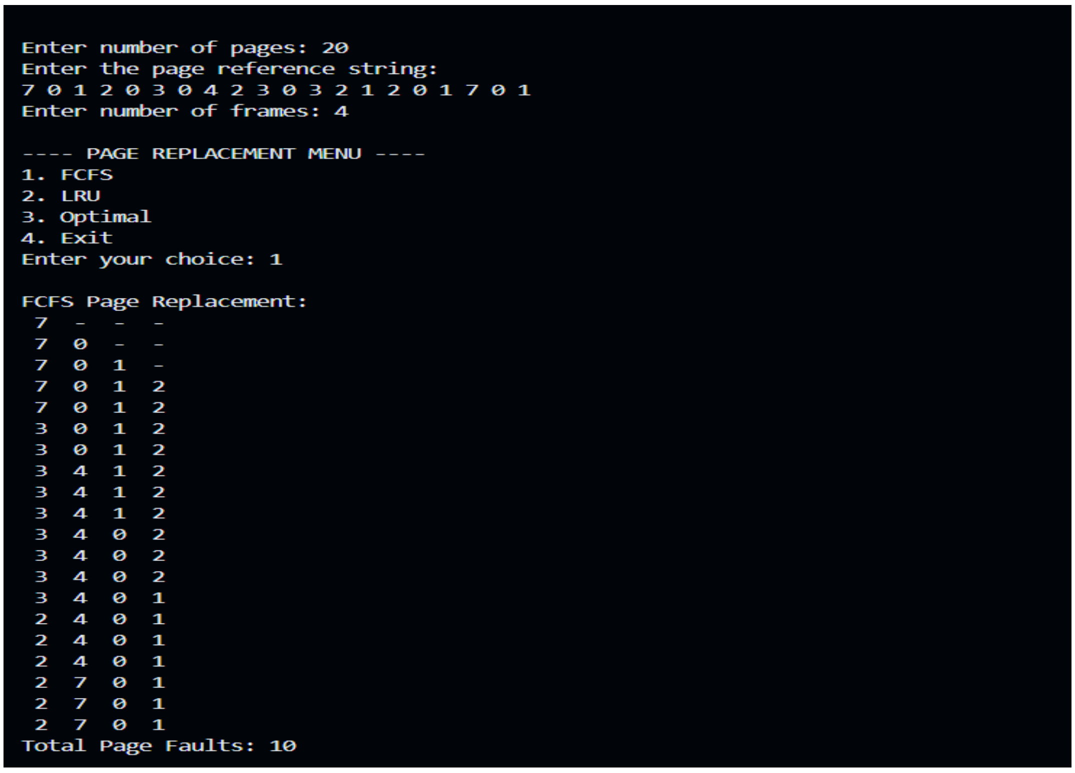
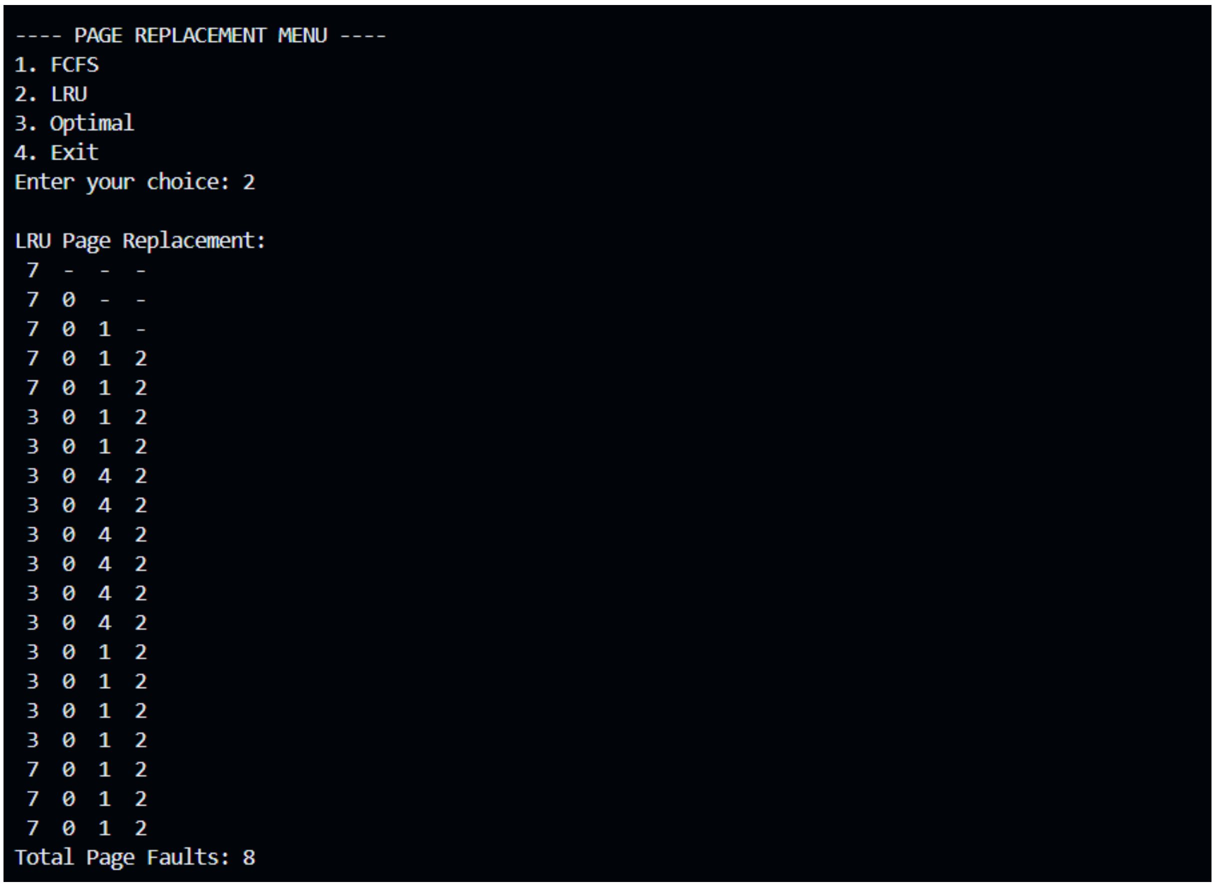
Assignment:-7

```
#include <stdio.h>
#define MAX 50

void printFrames(int frames[], int f) {
    for (int i = 0; i < f; i++) {
        if (frames[i] == -1)
            printf(" - ");
        else
            printf(" %d ", frames[i]);
    }
    printf("\n");
}

// ---------- FCFS ----------
void fcfs(int pages[], int n, int f) {
    int frames[MAX], faults = 0, i, j, k = 0, found;

    for (i = 0; i < f; i++) frames[i] = -1;

    printf("\nFCFS Page Replacement:\n");

    for (i = 0; i < n; i++) {
        found = 0;
        for (j = 0; j < f; j++) {
            if (frames[j] == pages[i]) {
                found = 1;
                break;
            }
        }

        if (!found) {
            frames[k] = pages[i];
            k = (k + 1) % f;
            faults++;
        }

        printFrames(frames, f);
    }

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

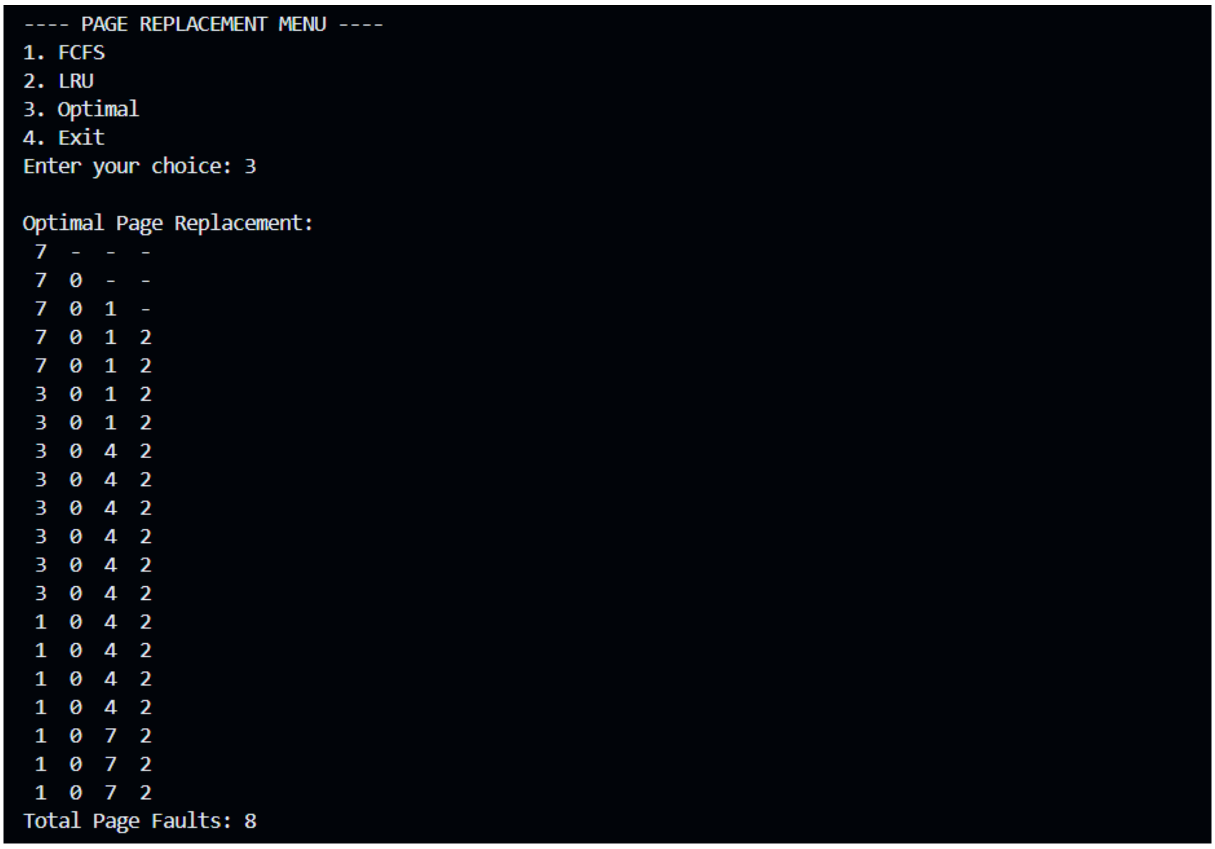
```c
}

// ---------- LRU ----------
int findLRU(int time[], int f) {
    int min = time[0], pos = 0;
    for (int i = 1; i < f; i++) {
        if (time[i] < min) {
            min = time[i];
            pos = i;
        }
    }
    return pos;
}

void lru(int pages[], int n, int f) {
    int frames[MAX], time[MAX], faults = 0, counter = 0, i, j, found, pos;

    for (i = 0; i < f; i++) frames[i] = -1;

    printf("\nLRU Page Replacement:\n");

    for (i = 0; i < n; i++) {
        found = 0;
        for (j = 0; j < f; j++) {
            if (frames[j] == pages[i]) {
                counter++;
                time[j] = counter;
                found = 1;
                break;
            }
        }

        if (!found) {
            pos = -1;
            for (j = 0; j < f; j++) {
                if (frames[j] == -1) {
                    pos = j;
                    break;
                }
            }
            if (pos == -1)
                pos = findLRU(time, f);
            counter++;
            frames[pos] = pages[i];
            time[pos] = counter;
            faults++;
        }

        printFrames(frames, f);
    }

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

// ---------- Optimal ----------
int predict(int pages[], int frames[], int n, int f, int index) {
    int res = -1, farthest = index;
    for (int i = 0; i < f; i++) {
        int j;
        for (j = index; j < n; j++) {
            if (frames[i] == pages[j]) {
                if (j > farthest) {
                    farthest = j;
                    res = i;
                }
                break;
            }
        }
        if (j == n) return i; // Not used again
    }
    return (res == -1) ? 0 : res;
}

void optimal(int pages[], int n, int f) {
    int frames[MAX], faults = 0, i, j, found;

    for (i = 0; i < f; i++) frames[i] = -1;

    printf("\nOptimal Page Replacement:\n");

    for (i = 0; i < n; i++) {
        found = 0;
        for (j = 0; j < f; j++) {
            if (frames[j] == pages[i]) {
                found = 1;
                break;
            }
        }

        if (!found) {
            int pos = -1;
            for (j = 0; j < f; j++) {
                if (frames[j] == -1) {
                    pos = j;
                    break;
                }
            }
            if (pos == -1)
                pos = predict(pages, frames, n, f, i + 1);

            frames[pos] = pages[i];
            faults++;
        }

        printFrames(frames, f);
    }

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

// ---------- MAIN MENU ----------
int main() {
    int pages[MAX], n, f, choice;

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

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

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

    do {
        printf("\n---- PAGE REPLACEMENT MENU ----\n");
        printf("1. FCFS\n2. LRU\n3. Optimal\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                fcfs(pages, n, f);
                break;
            case 2:
                lru(pages, n, f);
                break;
            case 3:
                optimal(pages, n, f);
                break;
            case 4:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice! Try again.\n");
        }
    } while (choice != 4);

    return 0;
}
```
OUTPUT



```
Enter number of pages: 20
Enter the page reference string:
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Enter number of frames: 4

---- PAGE REPLACEMENT MENU ----
1. FCFS
2. LRU
3. Optimal
4. Exit
Enter your choice: 1

FCFS Page Replacement:
 7  -  -  -
 7  0  -  -
 7  0  1  -
 7  0  1  2
 7  0  1  2
 3  0  1  2
 3  0  1  2
 3  4  1  2
 3  4  1  2
 3  4  1  2
 3  4  0  2
 3  4  0  2
 3  4  0  2
 3  4  0  1
 2  4  0  1
 2  4  0  1
 2  4  0  1
 2  7  0  1
 2  7  0  1
 2  7  0  1
Total Page Faults: 10
```



```
---- PAGE REPLACEMENT MENU ----
1. FCFS
2. LRU
3. Optimal
4. Exit
Enter your choice: 2

LRU Page Replacement:
 7  -  -  -
 7  0  -  -
 7  0  1  -
 7  0  1  2
 7  0  1  2
 3  0  1  2
 3  0  1  2
 3  0  4  2
 3  0  4  2
 3  0  4  2
 3  0  4  2
 3  0  4  2
 3  0  4  2
 3  0  1  2
 3  0  1  2
 3  0  1  2
 3  0  1  2
 7  0  1  2
 7  0  1  2
 7  0  1  2
Total Page Faults: 8
```



```
---- PAGE REPLACEMENT MENU ----
1. FCFS
2. LRU
3. Optimal
4. Exit
Enter your choice: 3

Optimal Page Replacement:
 7  -  -  -
 7  0  -  -
 7  0  1  -
 7  0  1  2
 7  0  1  2
 3  0  1  2
 3  0  1  2
 3  0  4  2
 3  0  4  2
 3  0  4  2
 3  0  4  2
 3  0  4  2
 3  0  4  2
 1  0  4  2
 1  0  4  2
 1  0  4  2
 1  0  4  2
 1  0  7  2
 1  0  7  2
 1  0  7  2
Total Page Faults: 8
```