OS Assignment:-6

```
#include <stdio.h>

int isPresent(int page, int f[], int n) {
    for (int i = 0; i < n; i++)
        if (f[i] == page)
            return 1;
    return 0;
}

int predict(int ref[], int f[], int n, int m, int index) {
    int res = -1, farthest = index;
    for (int i = 0; i < n; i++) {
        int j;
        for (j = index; j < m; j++) {
            if (f[i] == ref[j]) {
                if (j > farthest) {
                    farthest = j;
                    res = i;
                }
                break;
            }
        }
        if (j == m)
            return i;
    }
    return (res == -1) ? 0 : res;
}

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

void algoFCFS(int ref[], int n, int fsize) {
    int f[fsize], front = 0, pf = 0;
    for (int i = 0; i < fsize; i++) f[i] = -1;

    printf("\nRef\tFrames\t\tStatus\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t", ref[i]);
        if (!isPresent(ref[i], f, fsize)) {
            f[front] = ref[i];
            front = (front + 1) % fsize;
            pf++;
            showFrames(f, fsize);
            printf("\tFAULT\n");
        } else {
            showFrames(f, fsize);
            printf("\tHIT\n");
        }
    }
    printf("Total Faults (FCFS): %d\n", pf);
}

void algoLRU(int ref[], int n, int fsize) {
    int f[fsize], time[fsize], pf = 0;
    for (int i = 0; i < fsize; i++) {
        f[i] = -1;
        time[i] = -1;
    }

    printf("\nRef\tFrames\t\tStatus\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t", ref[i]);
        int found = 0;
        for (int j = 0; j < fsize; j++) {
            if (f[j] == ref[i]) {
                found = 1;
                time[j] = i;
                break;
            }
        }
        if (!found) {
            int lru = 0;
            for (int j = 1; j < fsize; j++) {
                if (time[j] < time[lru])
                    lru = j;
            }
            f[lru] = ref[i];
            time[lru] = i;
            pf++;
            showFrames(f, fsize);
            printf("\tFAULT\n");
        } else {
            showFrames(f, fsize);
            printf("\tHIT\n");
        }
    }
    printf("Total Faults (LRU): %d\n", pf);
}

void algoOptimal(int ref[], int n, int fsize) {
    int f[fsize], pf = 0;
    for (int i = 0; i < fsize; i++) f[i] = -1;

    printf("\nRef\tFrames\t\tStatus\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t", ref[i]);
        if (!isPresent(ref[i], f, fsize)) {
            int j;
            for (j = 0; j < fsize; j++) {
                if (f[j] == -1) {
                    f[j] = ref[i];
                    break;
                }
            }
            if (j == fsize) {
                int pos = predict(ref, f, fsize, n, i + 1);
                f[pos] = ref[i];
            }
            pf++;
            showFrames(f, fsize);
            printf("\tFAULT\n");
        } else {
            showFrames(f, fsize);
            printf("\tHIT\n");
        }
    }
    printf("Total Faults (Optimal): %d\n", pf);
}

int main() {
    int n, fsize, choice;
    printf("Enter length of reference string: ");
    scanf("%d", &n);
    int ref[n];
    printf("Enter reference string: ");
    for (int i = 0; i < n; i++) scanf("%d", &ref[i]);
    printf("Enter number of frames: ");
    scanf("%d", &fsize);

    while (1) {
        printf("\n=== Page Replacement Menu ===\n");
        printf("1. FCFS\n2. LRU\n3. Optimal\n4. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: algoFCFS(ref, n, fsize); break;
            case 2: algoLRU(ref, n, fsize); break;
            case 3: algoOptimal(ref, n, fsize); break;
            case 4: return 0;
            default: printf("Invalid option!\n");
        }
    }
}
```

OUTPUT:-

```
Enter length of reference string: 12
Enter reference string: 1 3 0 3 5 6 3 2 6 3 2 1
Enter number of frames: 4

=== Page Replacement Menu ===
1. FCFS
2. LRU
3. Optimal
4. Exit
Enter choice: 1

Ref     Frames          Status
1       1 - - -   FAULT
3       1 3 - -   FAULT
0       1 3 0 -   FAULT
3       1 3 0 -   HIT
5       1 3 0 5   FAULT
6       6 3 0 5   FAULT
3       6 3 0 5   HIT
2       6 2 0 5   FAULT
6       6 2 0 5   HIT
3       6 2 3 5   FAULT
2       6 2 3 5   HIT
1       1 2 3 5   FAULT
Total Faults (FCFS): 8

=== Page Replacement Menu ===
1. FCFS
2. LRU
3. Optimal
4. Exit
Enter choice: 2

Ref     Frames          Status
1       1 - - -   FAULT
3       1 3 - -   FAULT
0       1 3 0 -   FAULT
3       1 3 0 -   HIT
5       1 3 0 5   FAULT
6       6 3 0 5   FAULT
3       6 3 0 5   HIT
2       6 3 2 5   FAULT
6       6 3 2 5   HIT
3       6 3 2 5   HIT
2       6 3 2 5   HIT
1       1 3 2 5   FAULT
Total Faults (LRU): 7

=== Page Replacement Menu ===
1. FCFS
2. LRU
```

```
3. Optimal
4. Exit
Enter choice: 3

Ref     Frames          Status
1       1 - - -   FAULT
3       1 3 - -   FAULT
0       1 3 0 -   FAULT
3       1 3 0 -   HIT
5       1 3 0 5   FAULT
6       6 3 0 5   FAULT
3       6 3 0 5   HIT
2       6 3 2 5   FAULT
6       6 3 2 5   HIT
3       6 3 2 5   HIT
2       6 3 2 5   HIT
1       1 3 2 5   FAULT
Total Faults (Optimal): 7

=== Page Replacement Menu ===
1. FCFS
2. LRU
3. Optimal
4. Exit
Enter choice: 4
```