**Ex. No.: 11a)**

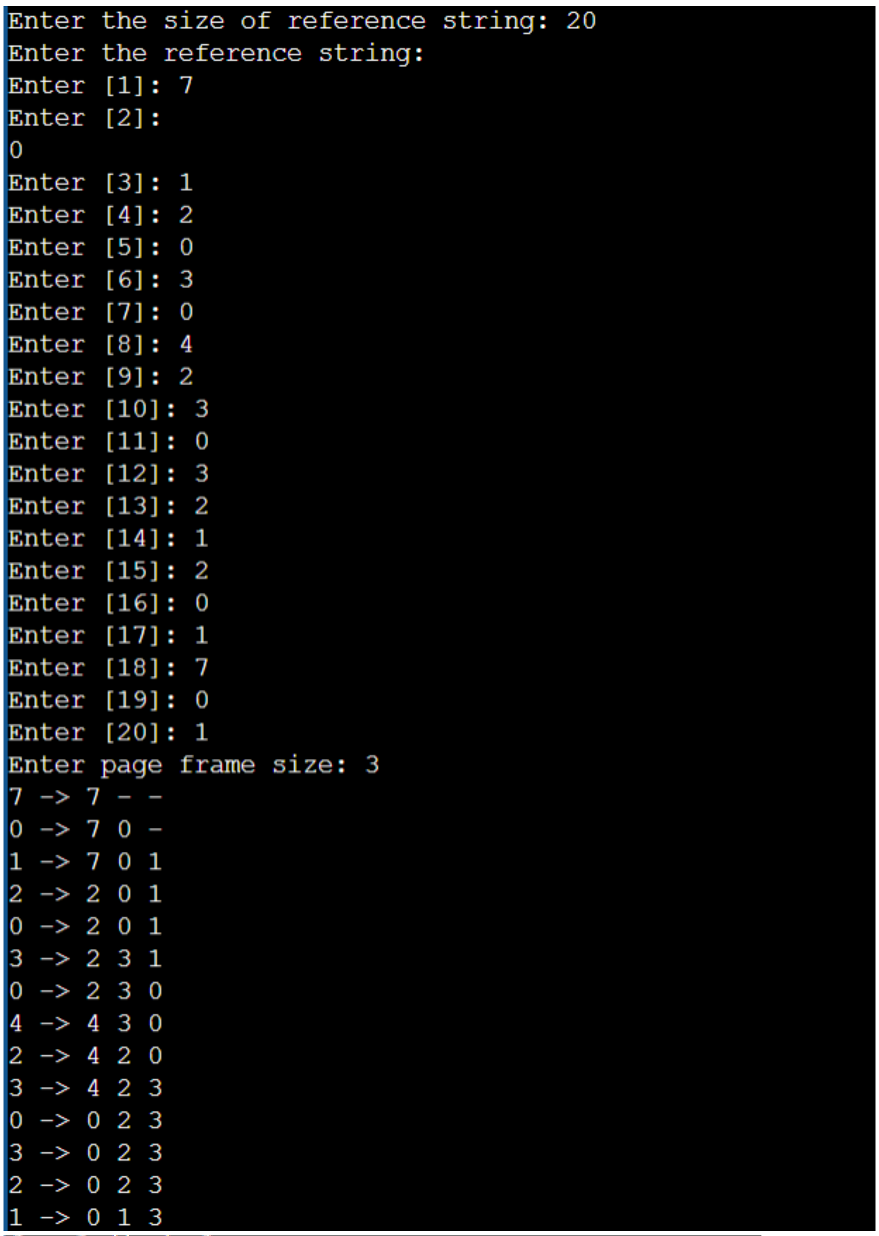
## FIFO PAGE REPLACEMENT

**Aim:**
To find out the number of page faults that occur using First-in First-out (FIFO) page replacement technique.

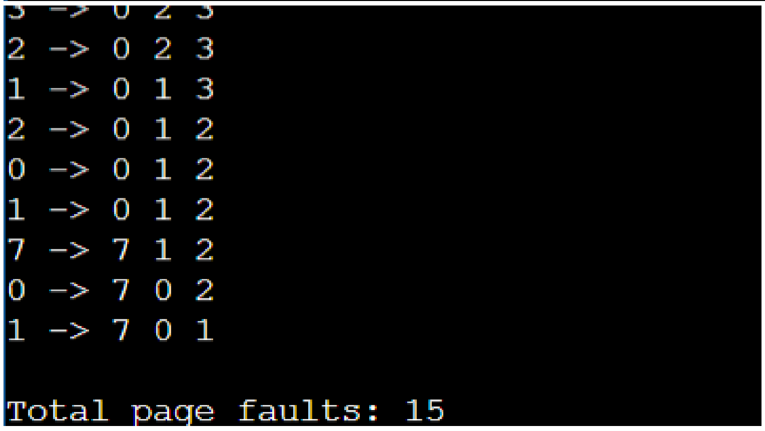
**Program code:**

```
#include <stdio.h>

#define MAX 50

int main() {
    int referenceString[MAX], pageFrame[MAX], pageFaults = 0;
    int frames, referenceLength;

    // Input reference string length and the string itself
    printf("Enter the size of reference string: ");
    scanf("%d", &referenceLength);

    printf("Enter the reference string:\n");
    for (int i = 0; i < referenceLength; i++) {
        printf("Enter [%d]: ", i + 1);
        scanf("%d", &referenceString[i]);
    }

    // Input number of frames
    printf("Enter page frame size: ");
    scanf("%d", &frames);

    // Initialize the page frame array to -1 (empty)
    for (int i = 0; i < frames; i++) {
        pageFrame[i] = -1;
    }

    // FIFO page replacement
    int front = 0;  // Points to the oldest page in the frame
    for (int i = 0; i < referenceLength; i++) {
        int page = referenceString[i];
        int found = 0;

        // Check if page is already in frame
        for (int j = 0; j < frames; j++) {
            if (pageFrame[j] == page) {
                found = 1;
                break;
            }
        }

        // If the page is not found in frame, perform page replacement
        if (!found) {
            pageFrame[front] = page;
            pageFaults++;

            // Move the front pointer (FIFO replacement)
            front = (front + 1) % frames;
        }

        // Print the current state of the page frame
        printf("%d -> ", page);
        for (int j = 0; j < frames; j++) {
            if (pageFrame[j] != -1) {
                printf("%d ", pageFrame[j]);
            } else {
                printf("- ");
            }
        }
        printf("\n");
    }

    printf("\nTotal page faults: %d\n", pageFaults);

    return 0;
}
```

**OUTPUT :**



```
Enter the size of reference string: 20
Enter the reference string:
Enter [1]: 7
Enter [2]:
0
Enter [3]: 1
Enter [4]: 2
Enter [5]: 0
Enter [6]: 3
Enter [7]: 0
Enter [8]: 4
Enter [9]: 2
Enter [10]: 3
Enter [11]: 0
Enter [12]: 3
Enter [13]: 2
Enter [14]: 1
Enter [15]: 2
Enter [16]: 0
Enter [17]: 1
Enter [18]: 7
Enter [19]: 0
Enter [20]: 1
Enter page frame size: 3
7 -> 7 - -
0 -> 7 0 -
1 -> 7 0 1
2 -> 2 0 1
0 -> 2 0 1
3 -> 2 3 1
0 -> 2 3 0
4 -> 4 3 0
2 -> 4 2 0
3 -> 4 2 3
0 -> 0 2 3
3 -> 0 2 3
2 -> 0 2 3
1 -> 0 1 3
```



```
3 -> 0 2 3
2 -> 0 2 3
1 -> 0 1 3
2 -> 0 1 2
0 -> 0 1 2
1 -> 0 1 2
7 -> 7 1 2
0 -> 7 0 2
1 -> 7 0 1

Total page faults: 15
```

**Ex. No.: 11b)**

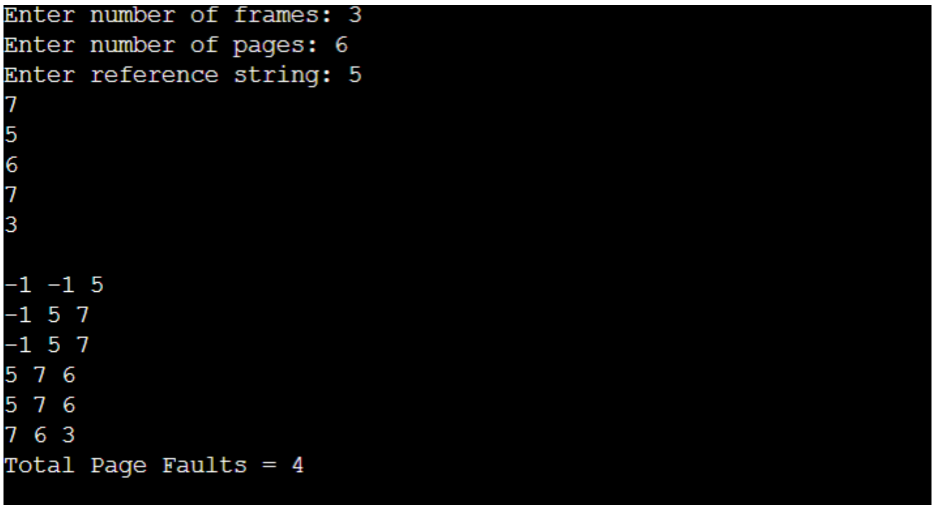
## LRU

**Aim:**
To write a c program to implement LRU page replacement algorithm.

**Program code :**

```
#include <stdio.h>

#define MAX 50

int main() {
    int referenceString[MAX], pageFrame[MAX], pageFaults = 0;
    int frames, referenceLength;

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

    // Input number of pages and the reference string
    printf("Enter number of pages: ");
    scanf("%d", &referenceLength);

    printf("Enter reference string: ");
    for (int i = 0; i < referenceLength; i++) {
        scanf("%d", &referenceString[i]);
    }

    // Initialize the page frame array to -1 (empty)
    for (int i = 0; i < frames; i++) {
        pageFrame[i] = -1;
    }

    // LRU page replacement
    for (int i = 0; i < referenceLength; i++) {
        int page = referenceString[i];
        int found = 0;

        // Check if the page is already in the frame
        for (int j = 0; j < frames; j++) {
            if (pageFrame[j] == page) {
                found = 1;
                break;
            }
        }
        // If the page is not found, replace the least recently used page
        if (!found) {
            // Shift pages in the frame to the left (LRU)
            for (int j = 0; j < frames - 1; j++) {
                pageFrame[j] = pageFrame[j + 1];
            }
            pageFrame[frames - 1] = page;
            pageFaults++;
        }

        // Print the current state of the page frame
        printf("\n");
        for (int j = 0; j < frames; j++) {
            printf("%d ", pageFrame[j]);
        }
    }

    printf("\nTotal Page Faults = %d\n", pageFaults);

    return 0;
}
```

**OUTPUT :**



```
Enter number of frames: 3
Enter number of pages: 6
Enter reference string: 5
7
5
6
7
3

-1 -1 5
-1 5 7
-1 5 7
5 7 6
5 7 6
7 6 3
Total Page Faults = 4
```

**Ex. No.: 11c)**

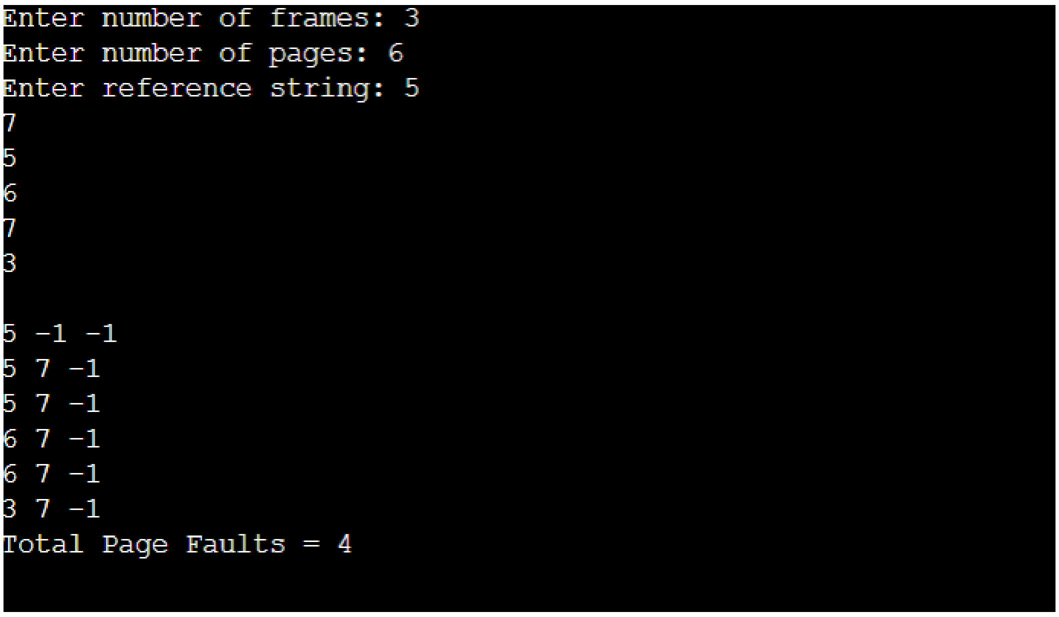
## Optimal

**Aim:**
To write a c program to implement Optimal page replacement algorithm.

**Program code :**

```
#include <stdio.h>

#define MAX 50

// Function to find the index of the page that will not be used for the longest time
int optimalPage(int referenceString[], int pageFrame[], int referenceLength, int frames, int
currentIndex) {
    int farthest = currentIndex;
    int replaceIndex = -1;

    for (int i = 0; i < frames; i++) {
        int j;
        for (j = currentIndex; j < referenceLength; j++) {
            if (pageFrame[i] == referenceString[j]) {
                if (j > farthest) {
                    farthest = j;
                    replaceIndex = i;
                }
                break;
            }
        }
        if (j == referenceLength) {
            return i; // If a page will not be used later, replace it
        }
    }
    return replaceIndex;
}

int main() {
    int referenceString[MAX], pageFrame[MAX], pageFaults = 0;
    int frames, referenceLength;

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

    // Input number of pages and the reference string
    printf("Enter number of pages: ");
    scanf("%d", &referenceLength);
    printf("Enter reference string: ");
    for (int i = 0; i < referenceLength; i++) {
        scanf("%d", &referenceString[i]);
    }

    // Initialize the page frame array to -1 (empty)
    for (int i = 0; i < frames; i++) {
        pageFrame[i] = -1;
    }

    // Optimal page replacement
    for (int i = 0; i < referenceLength; i++) {
        int page = referenceString[i];
        int found = 0;

        // Check if the page is already in the frame
        for (int j = 0; j < frames; j++) {
            if (pageFrame[j] == page) {
                found = 1;
                break;
            }
        }

        // If the page is not found, replace the optimal page
        if (!found) {
            int replaceIndex = optimalPage(referenceString, pageFrame, referenceLength,
frames, i);
            pageFrame[replaceIndex] = page;
            pageFaults++;
        }

        // Print the current state of the page frame
        printf("\n");
        for (int j = 0; j < frames; j++) {
            printf("%d ", pageFrame[j]);
        }
    }

    printf("\nTotal Page Faults = %d\n", pageFaults);

    return 0;
}
```

**OUTPUT :**



```
Enter number of frames: 3
Enter number of pages: 6
Enter reference string: 5
7
5
6
7
3

5 -1 -1
5 7 -1
5 7 -1
6 7 -1
6 7 -1
3 7 -1
Total Page Faults = 4
```