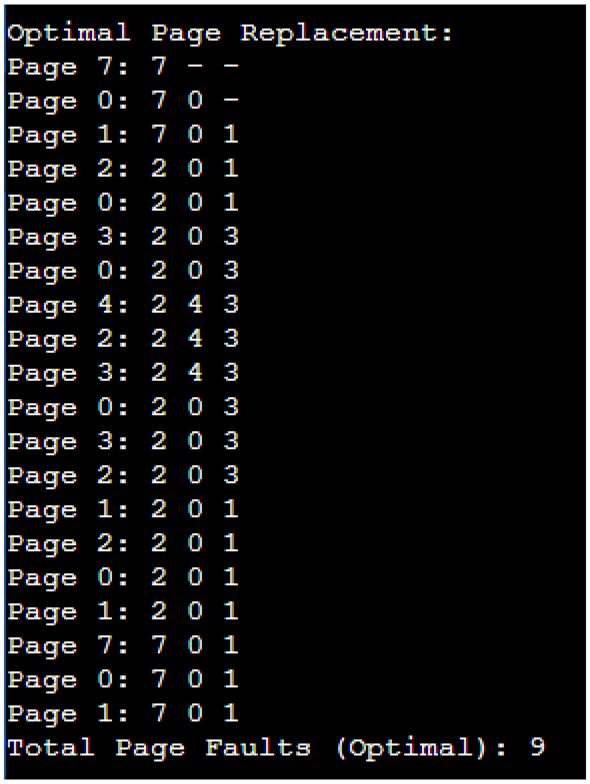
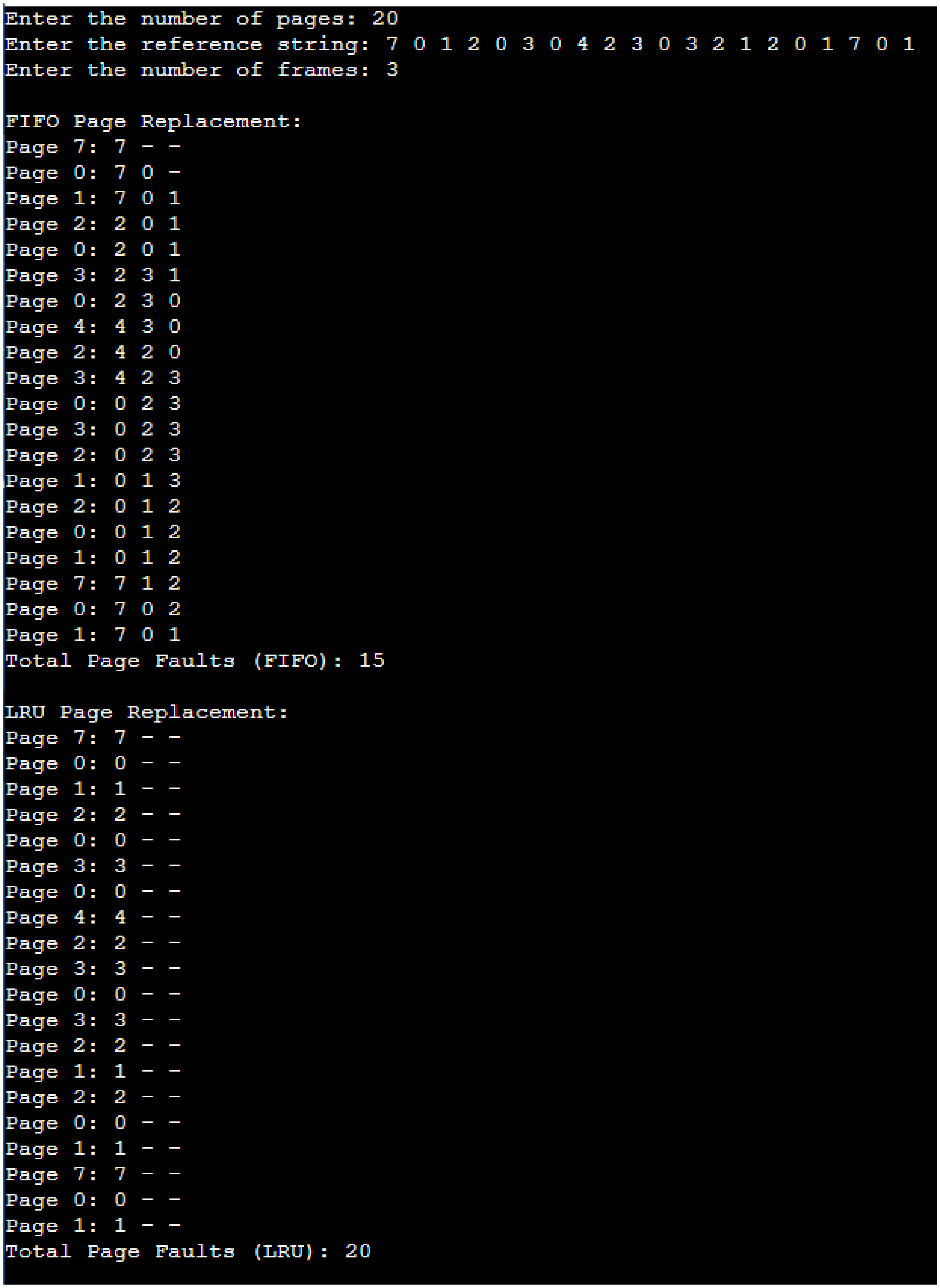
PAGE REPLACEMENT

```
#include <stdio.h>

#define MAX_FRAMES 3
#define MAX_PAGES 20

void fifo(int pages[], int n, int frames)
{
    int frame[frames];
    int front = 0, rear = 0;
    int page_faults = 0;
    for (int i = 0; i < frames; i++)
    {
        frame[i] = -1;
    }
    for (int i = 0; i < n; i++)
    {
        int found = 0;
        for (int j = 0; j < frames; j++)
        {
            if (frame[j] == pages[i])
            {
                found = 1;
                break;
            }
        }
        if (!found)
        {
            frame[rear] = pages[i];
            rear = (rear + 1) % frames;
            page_faults++;
        }
        printf("Page %d: ", pages[i]);
        for (int j = 0; j < frames; j++)
        {
            if (frame[j] == -1)
                printf("- ");
            else
                printf("%d ", frame[j]);
        }
        printf("\n");
    }
    printf("Total Page Faults (FIFO): %d\n", page_faults);
}

void lru(int pages[], int n, int frames) {
    int frame[frames];
    int page_faults = 0;
    int used[MAX_PAGES];
    for (int i = 0; i < frames; i++) {
        frame[i] = -1;
    }
    for (int i = 0; i < MAX_PAGES; i++) {
        used[i] = -1;
    }
    for (int i = 0; i < n; i++) {
        int found = 0;
        for (int j = 0; j < frames; j++) {
            if (frame[j] == pages[i]) {
                found = 1;
                used[frame[j]] = i;
                break;
            }
        }
        if (!found) {
            int min = 0;
            for (int j = 1; j < frames; j++) {
                if (used[frame[j]] < used[frame[min]]) {
                    min = j;
                }
            }
            frame[min] = pages[i];
            used[frame[min]] = i;
            page_faults++;
        }
        printf("Page %d: ", pages[i]);
        for (int j = 0; j < frames; j++) {
            if (frame[j] == -1)
                printf("- ");
            else
                printf("%d ", frame[j]);
        }
        printf("\n");
    }
    printf("Total Page Faults (LRU): %d\n", page_faults);
}
```

```
void optimal(int pages[], int n, int frames)
{
    int frame[frames];
    int page_faults = 0;

    for (int i = 0; i < frames; i++)
    {
        frame[i] = -1;
    }
    for (int i = 0; i < n; i++)
    {
        int found = 0;
        for (int j = 0; j < frames; j++)
        {
            if (frame[j] == pages[i])
            {
                found = 1;
                break;
            }
        }
        if (!found)
        {
            if (i < frames)
            {
                frame[i] = pages[i];
            }
            else
            {
                int max_dist = -1;
                int replace_page = -1;
                for (int j = 0; j < frames; j++)
                {
                    int dist = MAX_PAGES;
                    for (int k = i + 1; k < n; k++)
                    {
                        if (pages[k] == frame[j])
                        {
                            dist = k - i;
                            break;
                        }
                    }
                    if (dist > max_dist)
                    {
                        max_dist = dist;
                        replace_page = j;
                    }
                }
                frame[replace_page] = pages[i];
            }
            page_faults++;
        }
        printf("Page %d: ", pages[i]);
        for (int j = 0; j < frames; j++)
        {
            if (frame[j] == -1)
                printf("- ");
            else
                printf("%d ", frame[j]);
        }
        printf("\n");
    }

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

int main()
{
    int pages[MAX_PAGES];
    int n, frames;
    printf("Enter the number of pages: ");
    scanf("%d", &n);
    printf("Enter the reference string: ");
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &pages[i]);
    }
    printf("Enter the number of frames: ");
    scanf("%d", &frames);
    printf("\nFIFO Page Replacement:\n");
    fifo(pages, n, frames);
    printf("\nLRU Page Replacement:\n");
    lru(pages, n, frames);
    printf("\nOptimal Page Replacement:\n");
    optimal(pages, n, frames);
    return 0;
}
```

OUTPUT:


```
Optimal Page Replacement:
Page 7: 7 - -
Page 0: 7 0 -
Page 1: 7 0 1
Page 2: 2 0 1
Page 0: 2 0 1
Page 3: 2 0 3
Page 0: 2 0 3
Page 4: 2 4 3
Page 2: 2 4 3
Page 3: 2 4 3
Page 0: 2 0 3
Page 3: 2 0 3
Page 2: 2 0 3
Page 1: 2 0 1
Page 2: 2 0 1
Page 0: 2 0 1
Page 1: 2 0 1
Page 7: 7 0 1
Page 0: 7 0 1
Page 1: 7 0 1
Total Page Faults (Optimal): 9
```



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

FIFO Page Replacement:
Page 7: 7 - -
Page 0: 7 0 -
Page 1: 7 0 1
Page 2: 2 0 1
Page 0: 2 0 1
Page 3: 2 3 1
Page 0: 2 3 0
Page 4: 4 3 0
Page 2: 4 2 0
Page 3: 4 2 3
Page 0: 0 2 3
Page 3: 0 2 3
Page 2: 0 2 3
Page 1: 0 1 3
Page 2: 0 1 2
Page 0: 0 1 2
Page 1: 0 1 2
Page 7: 7 1 2
Page 0: 7 0 2
Page 1: 7 0 1
Total Page Faults (FIFO): 15

LRU Page Replacement:
Page 7: 7 - -
Page 0: 0 - -
Page 1: 1 - -
Page 2: 2 - -
Page 0: 0 - -
Page 3: 3 - -
Page 0: 0 - -
Page 4: 4 - -
Page 2: 2 - -
Page 3: 3 - -
Page 0: 0 - -
Page 3: 3 - -
Page 2: 2 - -
Page 1: 1 - -
Page 2: 2 - -
Page 0: 0 - -
Page 1: 1 - -
Page 7: 7 - -
Page 0: 0 - -
Page 1: 1 - -
Total Page Faults (LRU): 20
```