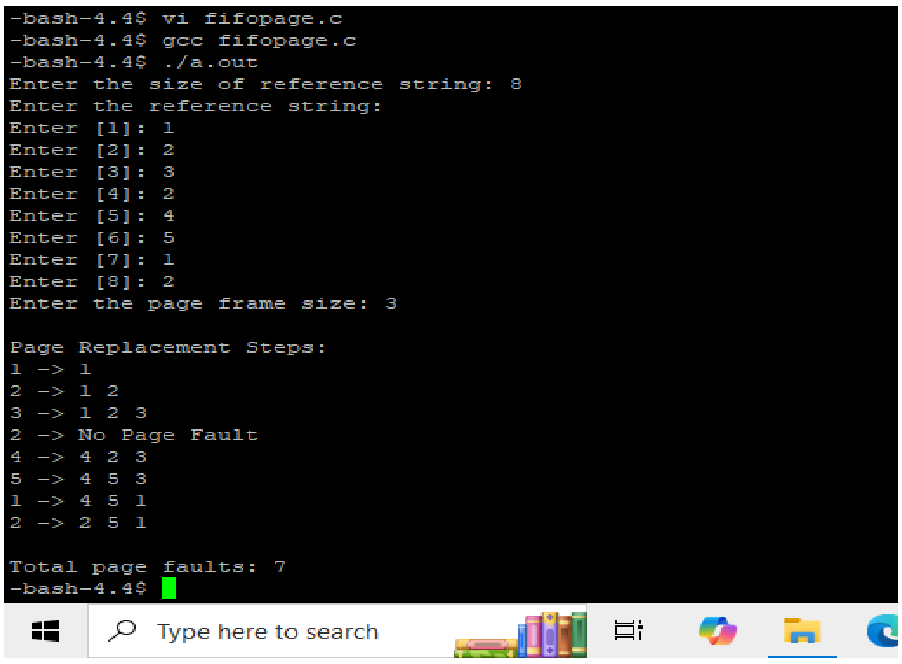
FIFO PAGE REPLACEMENT

Program:

```
#include <stdio.h>
int isPageInFrame(int page, int frame[], int size) {
    for (int i = 0; i < size; i++) {
        if (frame[i] == page)
            return 1;
    }
    return 0;
}
int main() {
    int reference[50], frame[10], ref_len, frame_size;
    int i, j, k = 0, faults = 0;
    printf("Enter the size of reference string: ");
    scanf("%d", &ref_len);
    printf("Enter the reference string:\n");
    for (i = 0; i < ref_len; i++) {
        printf("Enter [%d]: ", i + 1);
        scanf("%d", &reference[i]);
    }
    printf("Enter the page frame size: ");
    scanf("%d", &frame_size);

    for (i = 0; i < frame_size; i++)
```

```
        frame[i] = -1;  // initialize with -1

    printf("\nPage Replacement Steps:\n");
    for (i = 0; i < ref_len; i++) {
        printf("%d -> ", reference[i]);
        if (!isPageInFrame(reference[i], frame, frame_size)) {
            frame[k] = reference[i];
            k = (k + 1) % frame_size; // FIFO index cycle
            faults++;
            for (j = 0; j < frame_size; j++) {
                if (frame[j] != -1)
                    printf("%d ", frame[j]);
            }
            printf("\n");
        } else {
            printf("No Page Fault\n");
        }
    }
    printf("\nTotal page faults: %d\n", faults);
    return 0;
}
```

OUTPUT :



```
-bash-4.4$ vi fifopage.c
-bash-4.4$ gcc fifopage.c
-bash-4.4$ ./a.out
Enter the size of reference string: 8
Enter the reference string:
Enter [1]: 1
Enter [2]: 2
Enter [3]: 3
Enter [4]: 2
Enter [5]: 4
Enter [6]: 5
Enter [7]: 1
Enter [8]: 2
Enter the page frame size: 3

Page Replacement Steps:
1 -> 1
2 -> 1 2
3 -> 1 2 3
2 -> No Page Fault
4 -> 4 2 3
5 -> 4 5 3
1 -> 4 5 1
2 -> 2 5 1

Total page faults: 7
-bash-4.4$
```

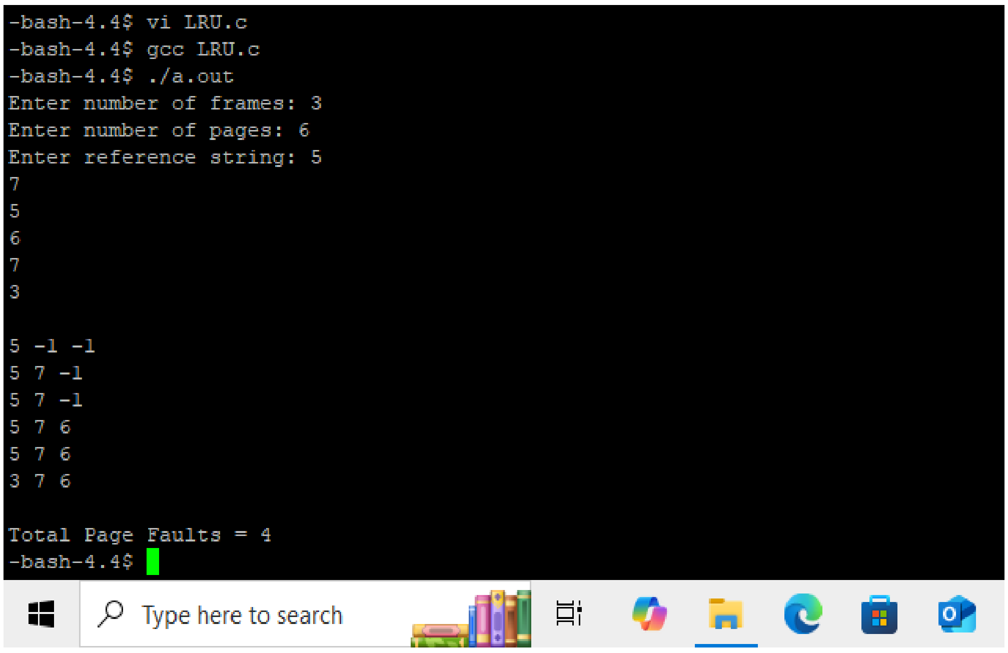
LRU

```
#include <stdio.h>
int findLRU(int time[], int n) {
    int i, min = time[0], pos = 0;
    for (i = 1; i < n; i++) {
        if (time[i] < min) {
            min = time[i];
            pos = i;
        }
    }
    return pos;
}
int main() {
    int frames[10], pages[30], counter[10];
    int n, f, i, j, k, pos, faults = 0, time = 0, flag1, flag2;
    printf("Enter number of frames: ");
    scanf("%d", &f);
    printf("Enter number of pages: ");
    scanf("%d", &n);
    printf("Enter reference string: ");
    for (i = 0; i < n; i++) {
        scanf("%d", &pages[i]);
    }
    for (i = 0; i < f; i++) {
        frames[i] = -1;
    }
    printf("\n");
```

```
for (i = 0; i < n; i++) {
    flag1 = flag2 = 0;
    for (j = 0; j < f; j++) {
        if (frames[j] == pages[i]) {
            time++;
            counter[j] = time;
            flag1 = flag2 = 1;
            break;
        }
    }
    if (flag1 == 0) {
        for (j = 0; j < f; j++) {
            if (frames[j] == -1) {
                time++;
                faults++;
                frames[j] = pages[i];
                counter[j] = time;
                flag2 = 1;
                break;
            }
        }
    }
    if (flag2 == 0) {
        pos = findLRU(counter, f);
        time++;
        faults++;
        frames[pos] = pages[i];
        counter[pos] = time;
    }
```

```
        for (j = 0; j < f; j++) {
            if (frames[j] != -1)
                printf("%d ", frames[j]);
            else
                printf("-1 ");
        }
        printf("\n");
    }
    printf("\nTotal Page Faults = %d\n", faults);
    return 0;
}
```

OUTPUT :



```
-bash-4.4$ vi LRU.c
-bash-4.4$ gcc LRU.c
-bash-4.4$ ./a.out
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
5 7 6
5 7 6
3 7 6

Total Page Faults = 4
-bash-4.4$
```

Optimal

Program :

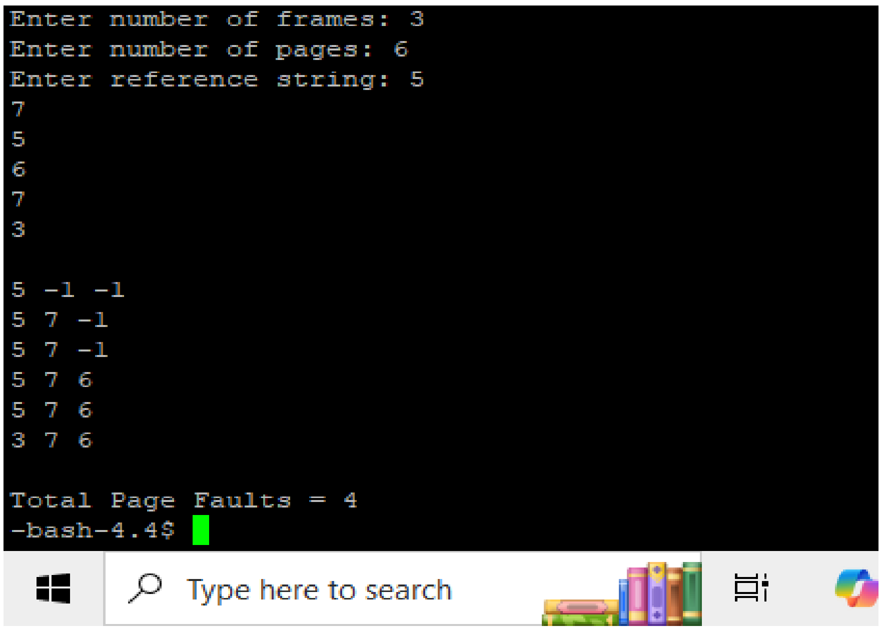
```
#include <stdio.h>


int findOptimal(int pages[], int frames[], int n, int f, int current) {
    int farthest = current, pos = -1;


    for (int i = 0; i < f; i++) {
        int j;
        for (j = current + 1; j < n; j++) {
            if (frames[i] == pages[j]) {
                if (j > farthest) {
                    farthest = j;
                    pos = i;
                }
                break;
            }
        }
        if (j == n) {
            return i; // If the page is not found in the future reference string
        }
    }
    return pos;
}


int main() {
    int frames[10], pages[30];
    int n, f, pageFaults = 0;
```

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

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

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

for (int i = 0; i < f; i++) {
    frames[i] = -1; // Initialize frames to -1 (empty)
}

printf("\n");

for (int i = 0; i < n; i++) {
    int found = 0;

    // Check if the page is already in memory
    for (int j = 0; j < f; j++) {
        if (frames[j] == pages[i]) {
            found = 1;
            break;
        }
    }
```

```
if (!found) {
    // If memory is full, find the optimal page to replace
    int replacePos = -1;

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

    // If no empty space, apply the optimal page replacement strategy
    if (!found) {
        replacePos = findOptimal(pages, frames, n, f, i);
        frames[replacePos] = pages[i];
        pageFaults++;
    }
}

// Display the current state of memory (frames)
for (int j = 0; j < f; j++) {
    if (frames[j] != -1)
        printf("%d ", frames[j]);
    else
        printf("-1 ");
}
printf("\n");
```

```
    }

    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
5 7 6
5 7 6
3 7 6

Total Page Faults = 4
-bash-4.4$ 
```