Ex. no: 11a

Name : Ashish P Shaji

Roll NO : 230701041

FIFO PAGE REPLACEMENT

Aim:

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

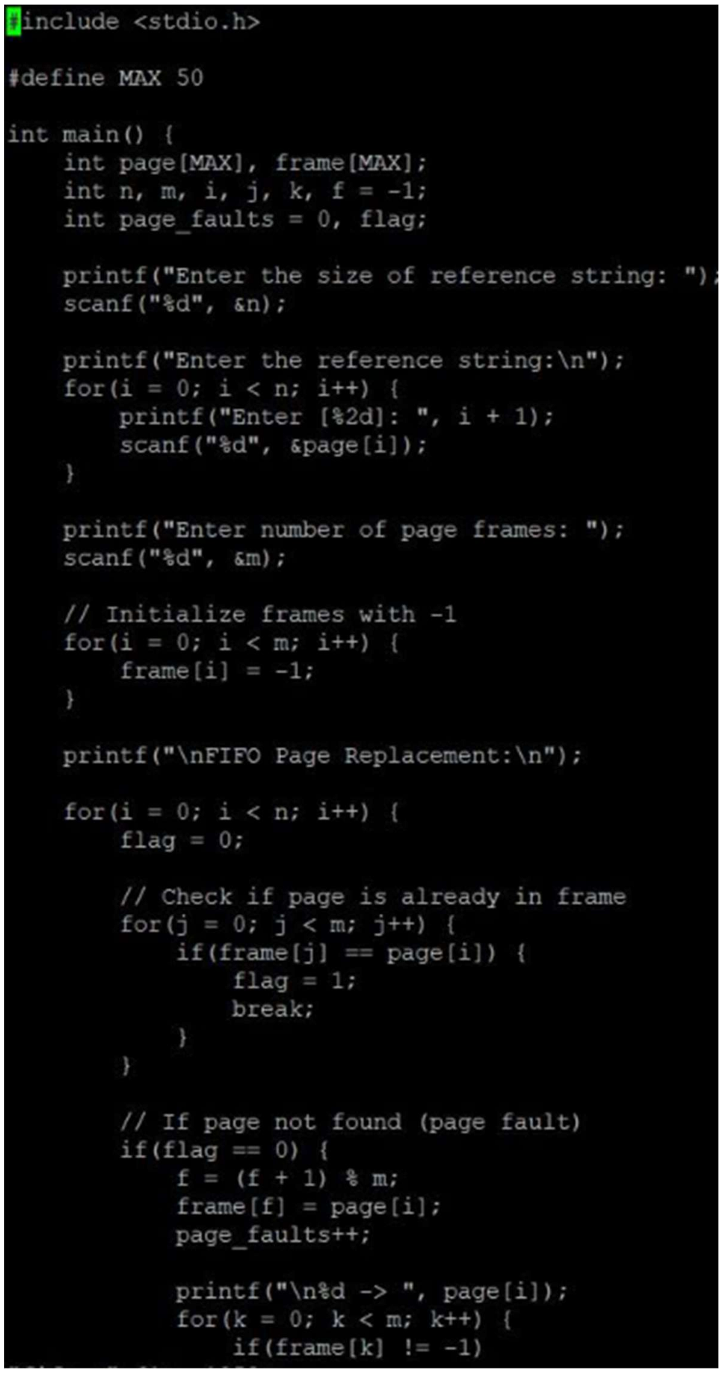
Algorithm:

1. Declare the size with respect to page length
2. Check the need of replacement from the page to memory
3. Check the need of replacement from old page to new page in memory 4.

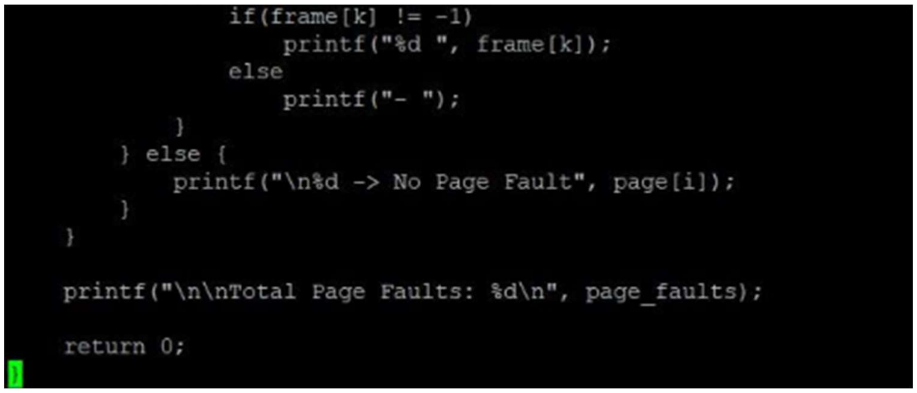
Form a queue to hold all pages

5. Insert the page require memory into the queue
6. Check for bad replacement and page fault
7. Get the number of processes to be inserted
8. Display the values

Program Code:



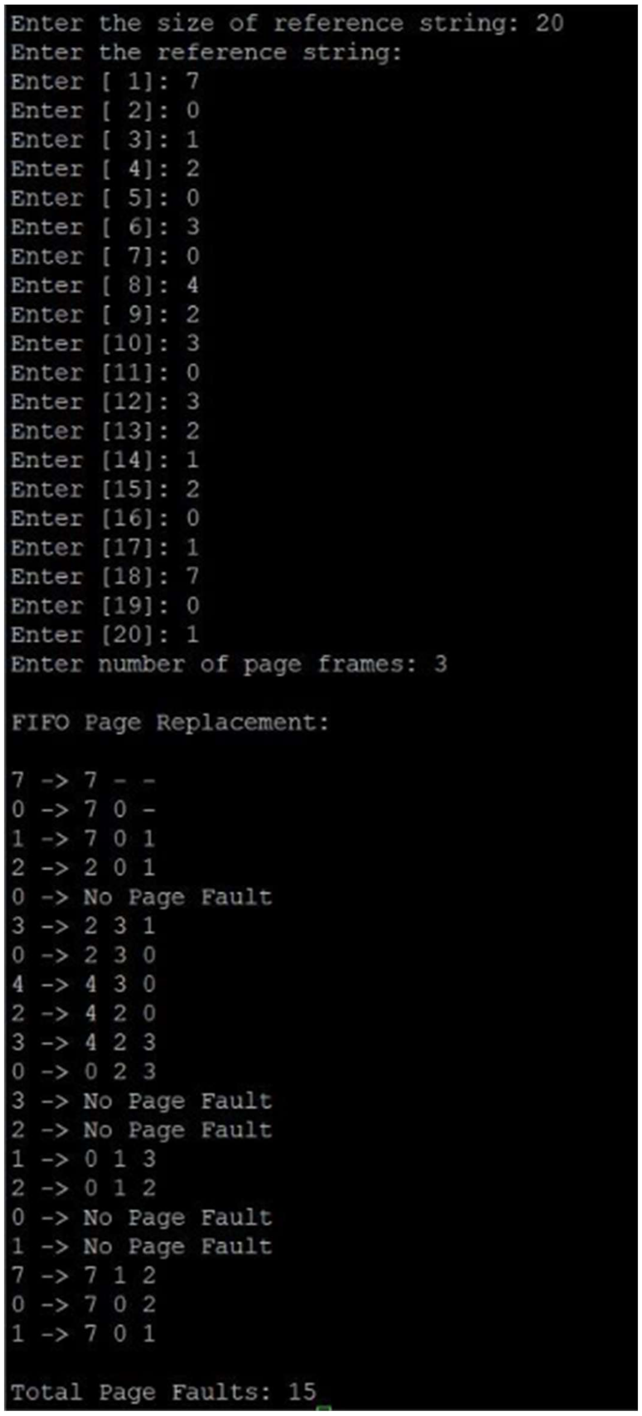
```
#include <stdio.h>

#define MAX 50

int main() {
    int page[MAX], frame[MAX];
    int n, m, i, j, k, f = -1;
    int page_faults = 0, flag;

    printf("Enter the size of reference string: ");
    scanf("%d", &n);

    printf("Enter the reference string:\n");
    for(i = 0; i < n; i++) {
        printf("Enter [%2d]: ", i + 1);
        scanf("%d", &page[i]);
    }

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

    // Initialize frames with -1
    for(i = 0; i < m; i++) {
        frame[i] = -1;
    }

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

    for(i = 0; i < n; i++) {
        flag = 0;

        // Check if page is already in frame
        for(j = 0; j < m; j++) {
            if(frame[j] == page[i]) {
                flag = 1;
                break;
            }
        }

        // If page not found (page fault)
        if(flag == 0) {
            f = (f + 1) % m;
            frame[f] = page[i];
            page_faults++;

            printf("\n%d -> ", page[i]);
            for(k = 0; k < m; k++) {
                if(frame[k] != -1)
```



```
                if(frame[k] != -1)
                    printf("%d ", frame[k]);
                else
                    printf("- ");
            }
        } else {
            printf("\n%d -> No Page Fault", page[i]);
        }
    }

    printf("\n\nTotal Page Faults: %d\n", page_faults);

    return 0;
}
```

OUTPUT :

LRU



```
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 number of page frames: 3

FIFO Page Replacement:

7 -> 7 - -
0 -> 7 0 -
1 -> 7 0 1
2 -> 2 0 1
0 -> No Page Fault
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 -> No Page Fault
2 -> No Page Fault
1 -> 0 1 3
2 -> 0 1 2
0 -> No Page Fault
1 -> No Page Fault
7 -> 7 1 2
0 -> 7 0 2
1 -> 7 0 1

Total Page Faults: 15
```

Ex. no: 11b

Name : Ashish P Shaji

Roll NO : 230701041

LRU

Aim:

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

Algorithm:

1: Start the process

2: Declare the size

3: Get the number of pages to be inserted

4: Get the value

5: Declare counter and stack

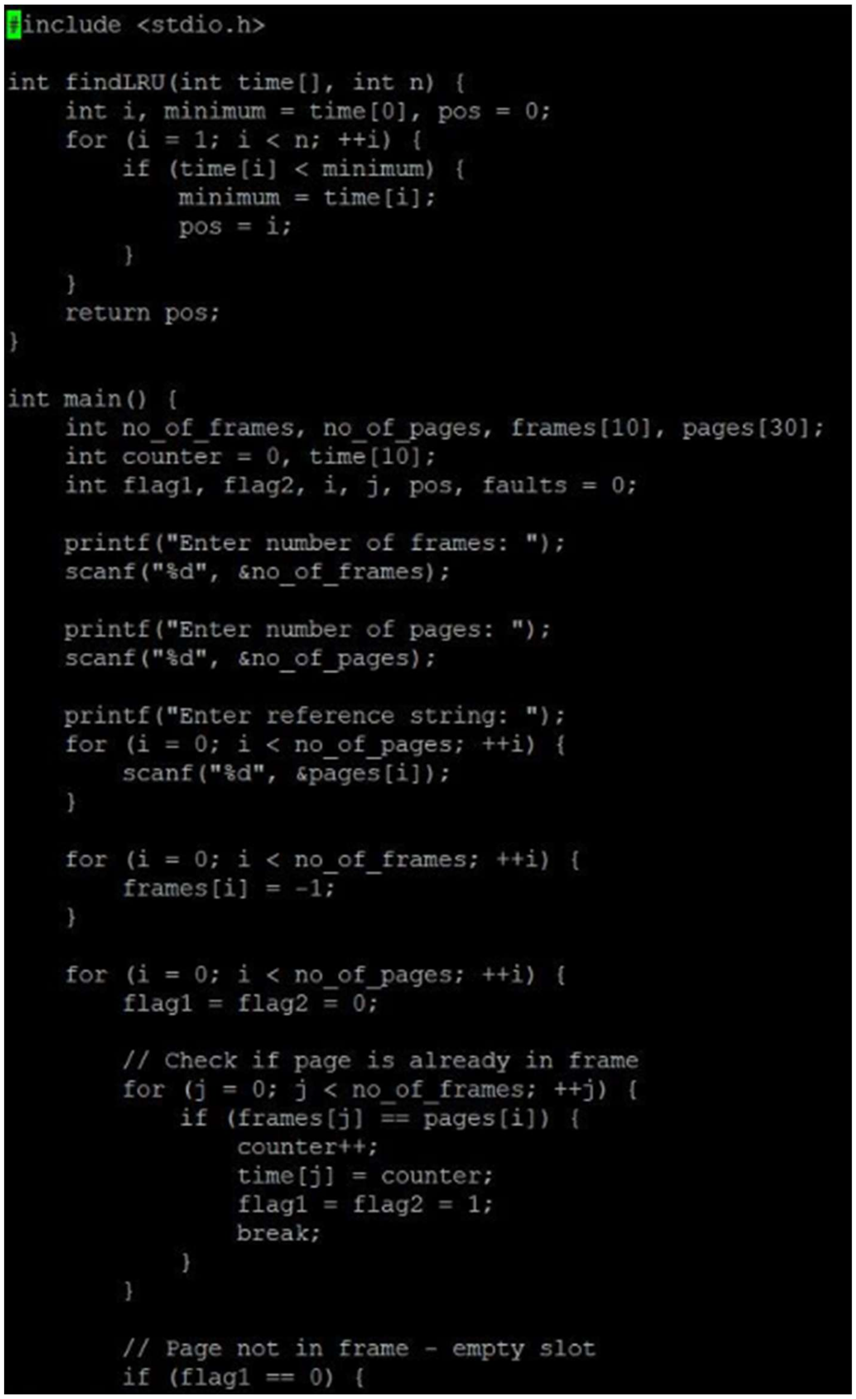
6: Select the least recently used page by counter value

7: Stack them according the selection.

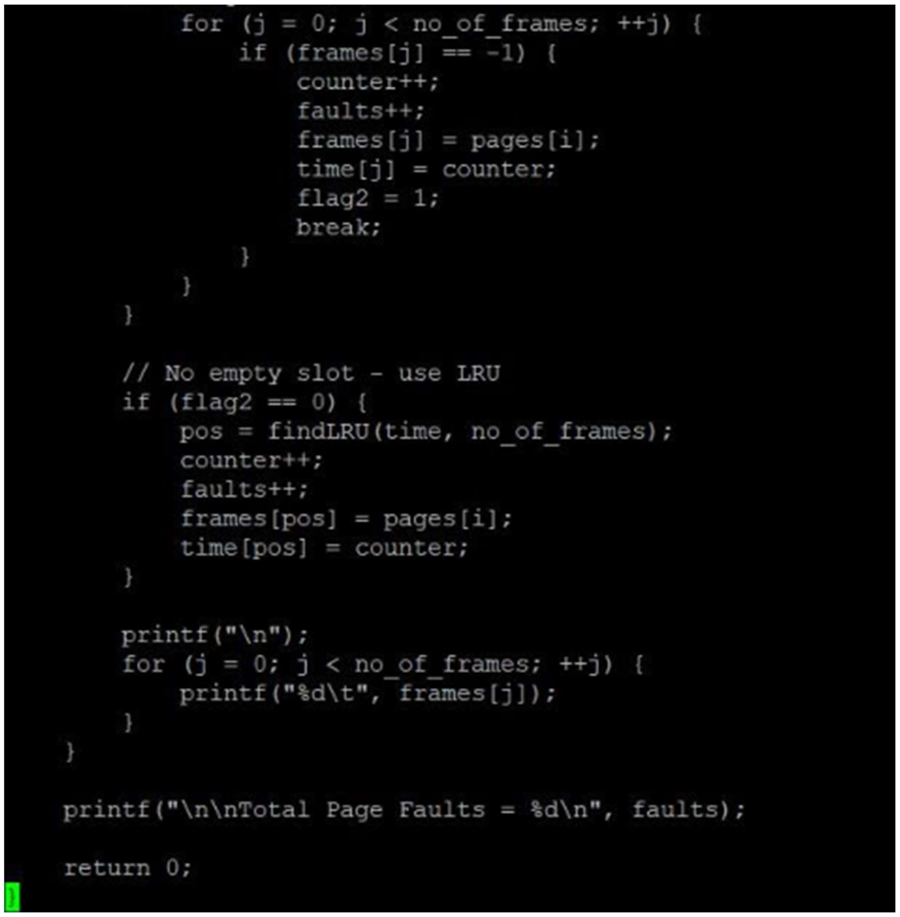
8: Display the values

9: Stop the process

Program Code:

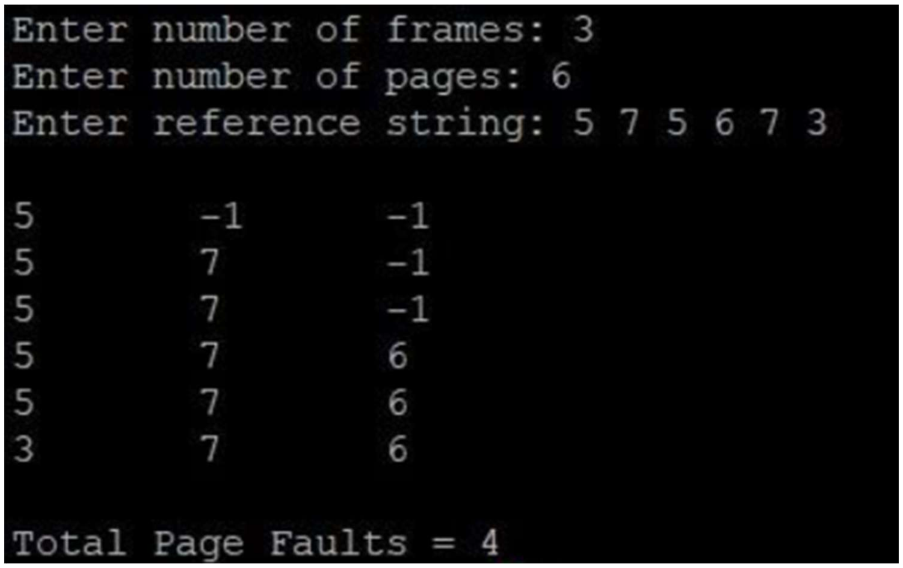


```c
#include <stdio.h>

int findLRU(int time[], int n) {
    int i, minimum = time[0], pos = 0;
    for (i = 1; i < n; ++i) {
        if (time[i] < minimum) {
            minimum = time[i];
            pos = i;
        }
    }
    return pos;
}

int main() {
    int no_of_frames, no_of_pages, frames[10], pages[30];
    int counter = 0, time[10];
    int flag1, flag2, i, j, pos, faults = 0;

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

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

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

    for (i = 0; i < no_of_frames; ++i) {
        frames[i] = -1;
    }

    for (i = 0; i < no_of_pages; ++i) {
        flag1 = flag2 = 0;

        // Check if page is already in frame
        for (j = 0; j < no_of_frames; ++j) {
            if (frames[j] == pages[i]) {
                counter++;
                time[j] = counter;
                flag1 = flag2 = 1;
                break;
            }
        }

        // Page not in frame - empty slot
        if (flag1 == 0) {
```



```
            for (j = 0; j < no_of_frames; ++j) {
                if (frames[j] == -1) {
                    counter++;
                    faults++;
                    frames[j] = pages[i];
                    time[j] = counter;
                    flag2 = 1;
                    break;
                }
            }
        }

        // No empty slot - use LRU
        if (flag2 == 0) {
            pos = findLRU(time, no_of_frames);
            counter++;
            faults++;
            frames[pos] = pages[i];
            time[pos] = counter;
        }

        printf("\n");
        for (j = 0; j < no_of_frames; ++j) {
            printf("%d\t", frames[j]);
        }
    }

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

    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
```

Ex. no: 11c

Name : Ashish P Shaji

Roll NO : 230701041

Optimal

Aim:

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

ALGORITHM:

1.Start the process

2.Declare the size

3.Get the number of pages to be inserted

4.Get the value

5.Declare counter and stack

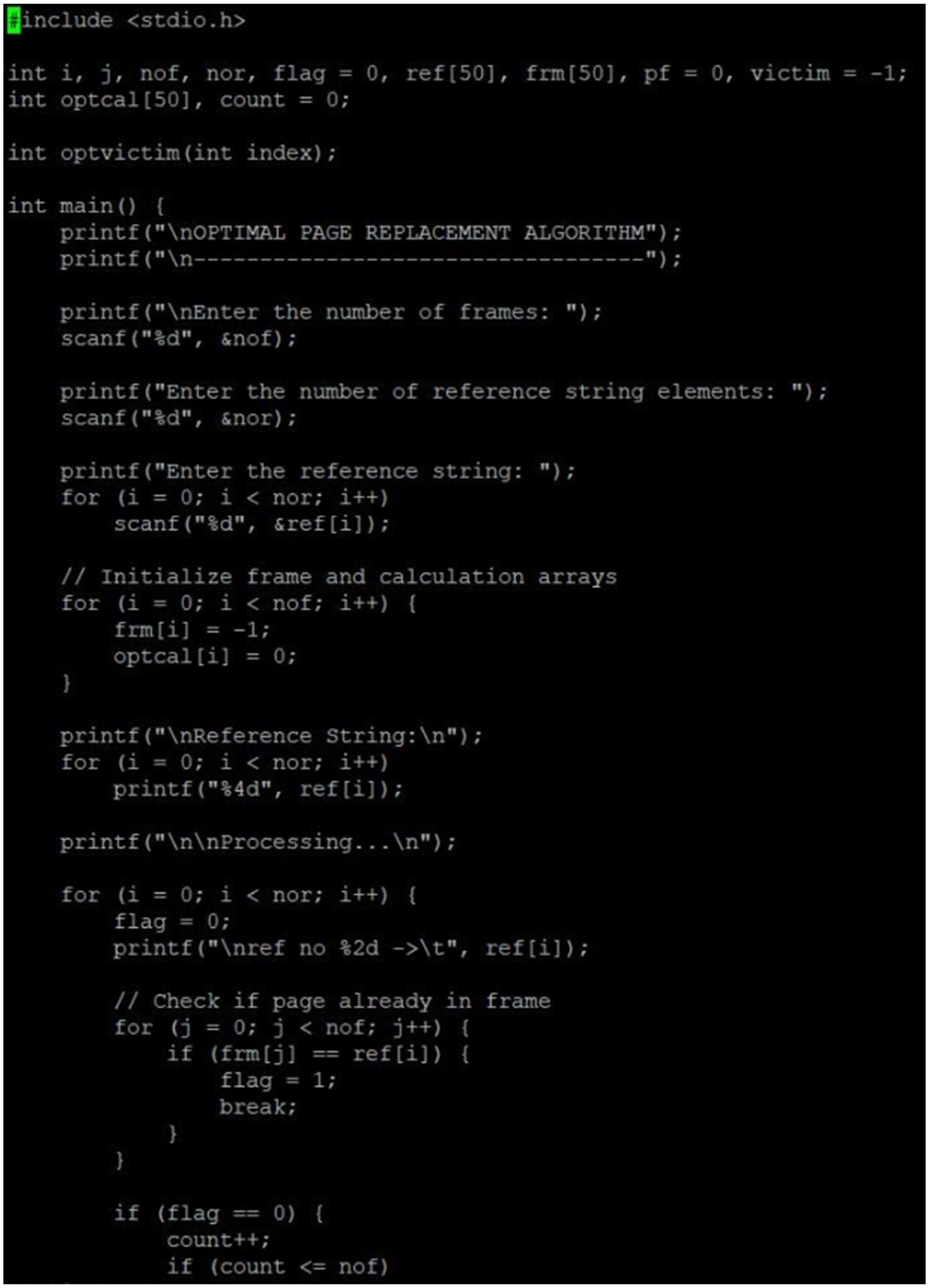
6.Select the least frequently used page by counter value

7.Stack them according the selection.

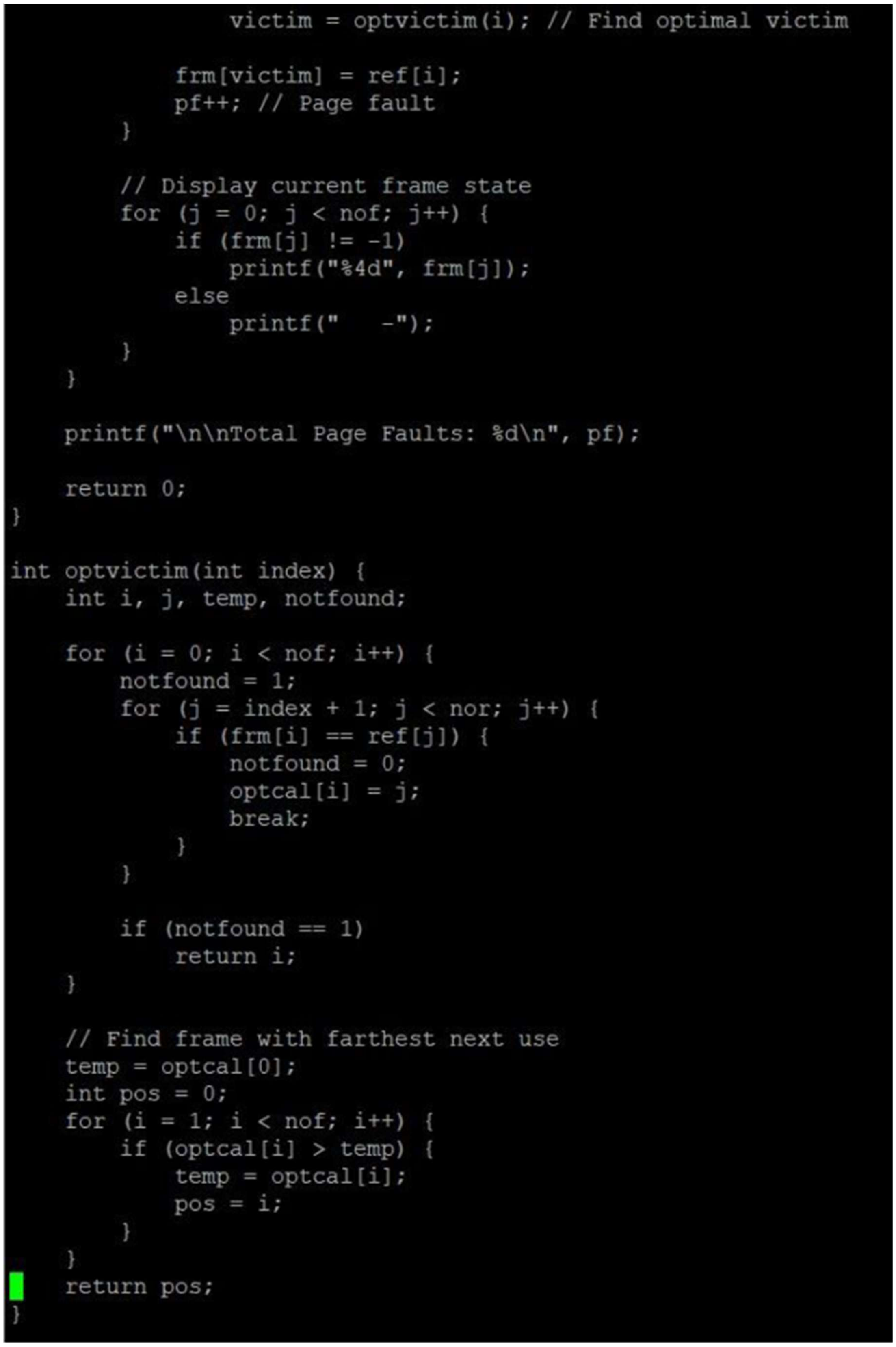
8.Display the values

9.Stop the process

PROGRAM:

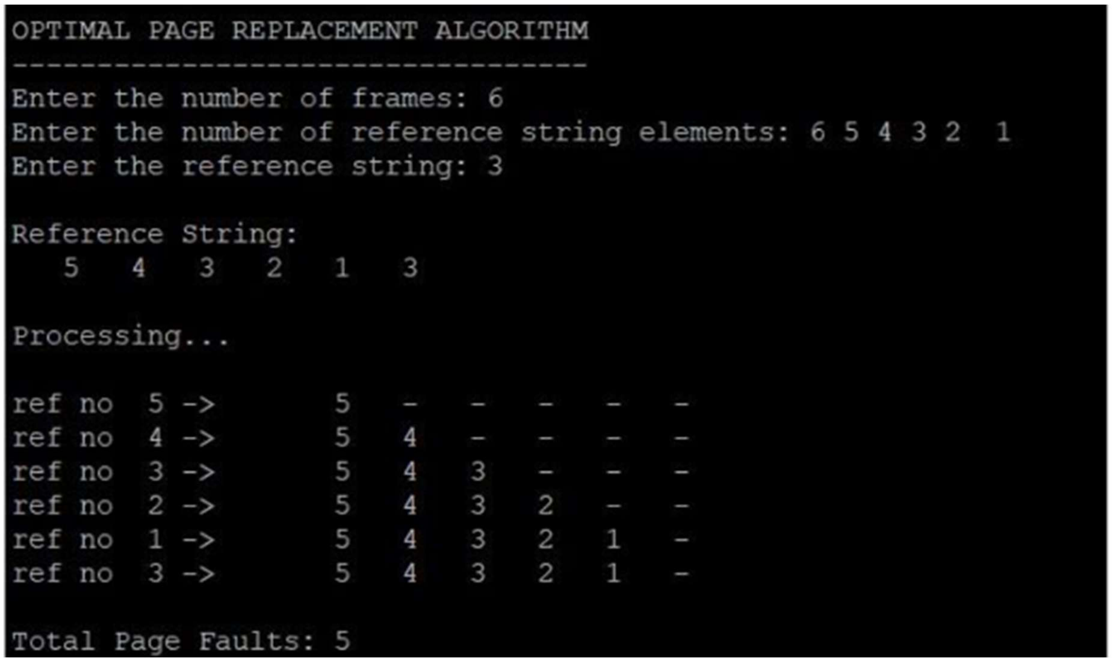


```c
#include <stdio.h>

int i, j, nof, nor, flag = 0, ref[50], frm[50], pf = 0, victim = -1;
int optcal[50], count = 0;

int optvictim(int index);

int main() {
    printf("\nOPTIMAL PAGE REPLACEMENT ALGORITHM");
    printf("\n--------------------------------");

    printf("\nEnter the number of frames: ");
    scanf("%d", &nof);

    printf("Enter the number of reference string elements: ");
    scanf("%d", &nor);

    printf("Enter the reference string: ");
    for (i = 0; i < nor; i++)
        scanf("%d", &ref[i]);

    // Initialize frame and calculation arrays
    for (i = 0; i < nof; i++) {
        frm[i] = -1;
        optcal[i] = 0;
    }

    printf("\nReference String:\n");
    for (i = 0; i < nor; i++)
        printf("%4d", ref[i]);

    printf("\n\nProcessing...\n");

    for (i = 0; i < nor; i++) {
        flag = 0;
        printf("\nref no %2d ->\t", ref[i]);

        // Check if page already in frame
        for (j = 0; j < nof; j++) {
            if (frm[j] == ref[i]) {
                flag = 1;
                break;
            }
        }

        if (flag == 0) {
            count++;
            if (count <= nof)
```



```
                victim = optvictim(i); // Find optimal victim

            frm[victim] = ref[i];
            pf++; // Page fault
        }

        // Display current frame state
        for (j = 0; j < nof; j++) {
            if (frm[j] != -1)
                printf("%4d", frm[j]);
            else
                printf("   -");
        }
    }

    printf("\n\nTotal Page Faults: %d\n", pf);

    return 0;
}

int optvictim(int index) {
    int i, j, temp, notfound;

    for (i = 0; i < nof; i++) {
        notfound = 1;
        for (j = index + 1; j < nor; j++) {
            if (frm[i] == ref[j]) {
                notfound = 0;
                optcal[i] = j;
                break;
            }
        }

        if (notfound == 1)
            return i;
    }

    // Find frame with farthest next use
    temp = optcal[0];
    int pos = 0;
    for (i = 1; i < nof; i++) {
        if (optcal[i] > temp) {
            temp = optcal[i];
            pos = i;
        }
    }
    return pos;
}
```

OUTPUT :



```
OPTIMAL PAGE REPLACEMENT ALGORITHM
----------------------------------
Enter the number of frames: 6
Enter the number of reference string elements: 6 5 4 3 2  1
Enter the reference string: 3

Reference String:
   5   4   3   2   1   3

Processing...

ref no  5 ->       5   -   -   -   -   -
ref no  4 ->       5   4   -   -   -   -
ref no  3 ->       5   4   3   -   -   -
ref no  2 ->       5   4   3   2   -   -
ref no  1 ->       5   4   3   2   1   -
ref no  3 ->       5   4   3   2   1   -

Total Page Faults: 5
```