PAGE REPLACEMENT FIFO

```
#include<stdio.h>
void main()
{
int i,j,n,a[50],frame[10],no,k,avail,count=0;
printf("\n ENTER THE NUMBER OF
PAGES:\n");
scanf("%d",&n);
        printf("\n ENTER THE PAGE
NUMBER :\n");
        for(i=1;i<=n;i++)
        scanf("%d",&a[i]);
        printf("\n ENTER THE NUMBER OF
FRAMES :");
        scanf("%d",&no);
for(i=0;i<no;i++)
        frame[i]= -1;
j=0;
printf("ref string\t page frames\n");
printf("_____
_____\n");
for(i=1;i<=n;i++)
  {
    printf("%d\t\t",a[i]);
    avail=0;
    for(k=0;k<no;k++)
 if(frame[k]==a[i])
 {
  for(k=0;k<no;k++)
   if(frame[k]!=-1)

printf("%d\t",frame[k]);

  avail=1;
 }
if (avail==0)
      {
           frame[j]=a[i];
           j=(j+1)%no;
           count++;
           for(k=0;k<no;k++)
  if(frame[k]!=-1)
                printf("%d\t",frame[k]);
    }
 printf("\n");
}
printf("Page Fault Is %d\n",count);
}
```

OUTPUT

```
administrator@administrator-Vostro-3800:~/
sslab$ gcc fifopage.c
administrator@administrator-Vostro-3800:~/
sslab$ ./a.out

 ENTER THE NUMBER OF PAGES:
10

 ENTER THE PAGE NUMBER :
4 7 6 1 7 6 1 2 7 2

 ENTER THE NUMBER OF FRAMES :3

ref string  page frames
_____
4  4
7  4 7
6  4 7 6
1  1 7 6
7  1 7 6
6  1 7 6
1  1 7 6
2  1 2 6
7  1 2 7
2  1 2 7
Page Fault Is 6
```

PAGE REPLACEMENT LRU

```
#include <stdio.h>
#include <limits.h>

int fr[3]; // Frame array

// Function to display the current frames
void display() {
    int i;
    printf("\n");
    for (i = 0; i < 3; i++) {
        if (fr[i] != -1)
            printf("\t%d", fr[i]);
        else
            printf("\t-");
    }
}

// Function to find the least recently used (LRU) page
int find_LRU(int timestamps[], int size) {
    int min_time = INT_MAX, index = -1;
    for (int i = 0; i < size; i++) {
        if (timestamps[i] < min_time) {
            min_time = timestamps[i];
            index = i;
        }
    }
    return index;
}

int main() {
    int p[12] = {2, 3, 2, 1, 5, 2, 4, 5, 3, 2, 5, 2}; // Page reference string
    int timestamps[3] = {0}, time = 0;
    int i, j, flag1, pf = 0, frsize = 3;

    // Initialize frame array
    for (i = 0; i < frsize; i++) {
        fr[i] = -1;
    }

    for (j = 0; j < 12; j++) {
        flag1 = 0;

        // Check if the page is already in the frame
        for (i = 0; i < frsize; i++) {
            if (fr[i] == p[j]) {
                flag1 = 1;
                timestamps[i] = ++time; // Update timestamp since it's used recently
                break;
            }
        }

        // If not found, page fault occurs
        if (flag1 == 0) {
            int replace_index = -1;

            // Look for an empty slot first
            for (i = 0; i < frsize; i++) {
                if (fr[i] == -1) { // Empty slot found
                    replace_index = i;
                    break;
                }
            }

            // If no empty slot, replace the LRU page
            if (replace_index == -1) {
                replace_index = find_LRU(timestamps, frsize);
            }

            fr[replace_index] = p[j];  // Replace page
            timestamps[replace_index] = ++time;
            pf++; // Increment page fault count
        }

        display();
    }

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

```
    return 0;
}
```

OUTPUT

```
        2       -       -
        2       3       -
        2       3       -
        2       3       1
        2       5       1
        2       5       1
        2       5       4
        2       5       4
        3       5       4
        3       5       2
        3       5       2
        3       5       2
Total Page Faults: 7
```

DISK SCHEDULING

```
#include<stdio.h>
#include<stdlib.h>
void main ()
{
        int queue
[20],n,head,i,j,k,seek=0,diff;
        float avg;
        printf("Enter the size of queue
request\n");
        scanf("%d",&n);
        printf("Enter the queue of disk
positions to be read\n");
        for(i=1;i<=n;i++)
        scanf("%d",&queue[i]);
        printf("Enter the initial head
position\n");
        scanf("%d",&head);
        queue[0]=head;
        for(j=0;j<=n-1;j++)
        {

diff=abs(queue[j+1]-queue[j]);
                seek+=diff;
printf("Disk head moves from %d to %d with
seek
%d\n",queue[j],queue[j+1],diff);
        }
        printf("Total seek time is
%d\n",seek);
        avg=seek/(float)n;
        printf("Average seek time is
%f\n",avg);
        }
```

OUTPUT

```
administrator@administrator-Vostro-3800:~/
sslab$ gcc fcfsdisk.c
administrator@administrator-Vostro-3800:~/
sslab$ ./a.out

Enter the size of queue request
8

Enter the queue of disk positions to be read

90 120 35 122 38 128 65 68

Enter the initial head position

50

Disk head moves from 50 to 90 with seek
40
Disk head moves from 90 to 120 with seek
30
Disk head moves from 120 to 35 with seek
85
Disk head moves from 35 to 122 with seek
87
Disk head moves from 122 to 38 with seek
84
Disk head moves from 38 to 128 with seek
90
Disk head moves from 128 to 65 with seek
63
Disk head moves from 65 to 68 with seek 3

Total seek time is 482
Average seek time is 60.250000
```