**EXERCISE-4**

Memory Management Scheme-II: **FIFO**

**Aim:**

To implement Memory Management Scheme-II: **FIFO**

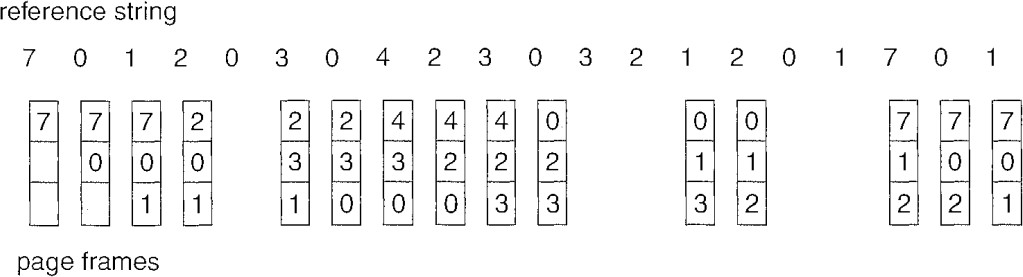
**Description:**

FIFO Page Replacement:

* The simplest page-replacement algorithm is a first-in, first-out (FIFO) algorithm.
* A FIFO replacement algorithm associates with each page the time when that page was brought into memory.

## When a page must be replaced, the oldest page is chosen.

* Notice that it is not strictly necessary to record the time when a page is brought in.
* We can create a FIFO queue to hold all pages in memory.
* We replace the page at the head of the queue.
* When a page is brought into memory, we insert it at the tail of the queue.



* Fifteen page faults are for the given reference string with 3 frames.
* The page-fault rate may increase as the number of allocated frames increases is called Belady’s anomaly.

**Program:**

#include<stdio.h>

struct page

{

int name,time,frequency;

}p[100];

struct frame

{

int name,time,frequency;

}f[100],temp;

void main()

{

int count=0,clock=0,nf,j,n,h,i,s,l,k=0,pagefault=0;

printf("enter the number of frames in physical memory:");

scanf("%d",&nf);

printf("enter the number of pages in memory reference string:");

scanf("%d",&n);

for(i=1;i<=nf;i++)

f[i].name=-1;

printf("enter the memory reference string:(0-9 only)\n");

for(i=1;i<=n;i++)

{

printf("m%d:",i);

scanf("%d",&p[i].name);

}

i=0;

while(count<n)

{

i++;

count++;

k=0;

if(i<=nf)

{

if(i==1)

{

clock=clock+1;

f[i].name=p[count].name;

f[i].time=clock;

f[i].frequency=1;

pagefault=pagefault+1;

}

else

{

for(j=1;j<=nf;j++)

{

if(f[j].name==p[count].name)

{

j=i;

k=1;

}

}

if(k==0)

{

clock=clock+1;

f[i].name=p[count].name;

f[i].time=clock;

f[i].frequency=1;

pagefault=pagefault+1;

}

if(k==1)

{

clock=clock+1;

i=i-1;

}

}

}

else

{

for(s=2;s<=nf;s++)

{

for(l=1;l<s;l++)

{

if(f[l].time>f[s].time)

{

temp=f[l];

f[l]=f[s];

f[s]=temp;

}

}

}

for(j=1;j<=nf;j++)

{

if(f[j].name==p[count].name)

{

j=i;

k=1;

}

}

if(k==0)

{

clock=clock+1;

f[1].name=p[count].name;

f[1].time=clock;

f[1].frequency=1;

pagefault=pagefault+1;

}

if(k==1)

{

clock=clock+1;

i=i-1;

}

}

printf("\nPages in frames are::");

if(k==1)

printf("page fault is not occured");

else

{

for(h=1;h<=nf;h++)

{

printf("%d\t",f[h].name);

}

}

};

printf("\n\nAt end of memory stream the pages in frames are::");

for(i=1;i<=nf;i++)

{

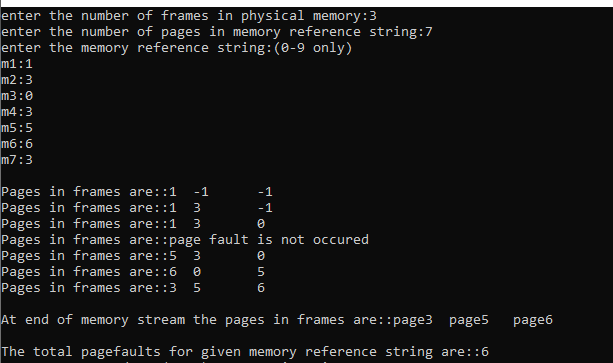
printf("page%d\t",f[i].name);

}

printf("\n\nThe total pagefaults for given memory reference string are::%d",pagefault);

}

**Output:**

****