**Ex.No: 7**

**FIFO Page Replacement**

**OBJECTIVE:** To Implement FIFO(First In First Out) Page Replacement Program

**DESCRIPTION:**

The simplest page-replacement algorithm is a FIFO algorithm. The first-in, first-out (FIFO) page replacement algorithm is a low-overhead algorithm that requires little bookkeeping on the part of the [operating system](https://en.wikipedia.org/wiki/Operating_system). The idea is obvious from the name – the operating system keeps track of all the pages in memory in a queue, with the most recent arrival at the back, and the oldest arrival in front. When a page needs to be replaced, the page at the front of the queue (the oldest page) is selected. While FIFO is cheap and intuitive, it performs poorly in practical application. Thus, it is rarely used in its unmodified form. This algorithm experiences [Bélády's anomaly](https://en.wikipedia.org/wiki/B%C3%A9l%C3%A1dy%27s_anomaly).

**Program:**

// FIFO page replacement algorithm.

#include<stdio.h>

int 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("\tref string\t page frames\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])

avail=1;

if (avail==0)

{

frame[j]=a[i];

j=(j+1)%no;

count++;

for(k=0;k<no;k++)

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

}

printf("\n");

}

printf(""No. of Page Faults: %d",count);

return 0;

}

**Ex.No: 8)**

**LRU Page Replacement**

**OBJECTIVE:** To implement LRU(Least Recently Used) Page Replacement Program

**DESCRIPTION:**

The least recently used (LRU) page replacement algorithmkeeps track of page usage over a short period of time. LRU works on the idea that pages that have been most heavily used in the past few instructions are most likely to be used heavily in the next few instructions too.

**Program:**

#include <stdio.h>

int main()

{

int i, j, k, f,max,p=10, pf=0, count[10], pageref[25], fp[10], n,flag[10];

printf("\n Enter the length of page reference string -- "); scanf("%d",&n);

printf("\n Enter the reference string -- ");

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

scanf("%d",&pageref[i]);

printf("\n Enter no. of frames -- ");

scanf("%d",&f);

for(i=0;i<f;i++)

{

fp[i]=-1;count[i]=0;flag[i]=0;

}

printf("\n The Page Replacement Process is -- \n");

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

{

for(k=0;k<f;k++)

{

if(count[k]==0)

{

fp[k]=pageref[i];

pf++;

count[k]=1;p=k;flag[k]=1; break;

}

else if(fp[k]==pageref[i]) //required page found

{

count[k]=1;p=k;flag[k]=1; break;

}

}

if(k==f) //LRU replacement

{

max=0;

for(j=0;j<f;j++)

{

if( count[j]>max)

{

max=count[j];

p=j;

}

}

fp[p]=pageref[i];

count[p]=1;

flag[p]=1;

pf++;

}

printf("Page ref is %d",pageref[i]);

for(j=0;j<f;j++)

{

if(j==p || count[j]==0)

continue;

count[j]=count[j]+1;

}

for(j=0;j<f;j++)

{

printf("\t%d ",fp[j]);

}

printf("Fault :%d",pf);

printf("\n");

}

printf("\n The number of Page Faults using LRU are %d",pf);

}

**Ex.No: 8b)**

**LFU Page Replacement**

**OBJECTIVE:** To implement LFU(Least Frequently Used) Page Replacement Program

**DESCRIPTION:**

The simplest method to employ an LFU algorithm is to assign a counter to every block that is loaded into the cache. Each time a reference is made to that block the counter is increased by one. When the cache reaches capacity and has a new block waiting to be inserted the system will search for the block with the lowest counter and remove it from the cache.

**Program:**

//LFU

#include<stdio.h>

int main()

{

int i, j, k, f,min,p, pf=0, count[10], pageref[25], fp[10], n;

printf("\n Enter the length of page reference string -- ");

scanf("%d",&n);

printf("\n Enter the reference string -- ");

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

scanf("%d",&pageref[i]);

printf("\n Enter no. of frames -- ");

scanf("%d",&f);

for(i=0;i<f;i++)

{

fp[i]=-1;

count[i]=0;

}

printf("\n The Page Replacement Process is -- \n");

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

{

for(k=0;k<f;k++)

{

if(fp[k]==pageref[i])

{

count[k]++;

break;

}

}

if(k==f)

{

min=100;

for(j=0;j<f;j++)

{

if( count[j]<min)

{

min=count[j];

p=j;

}

}

fp[p]=pageref[i];

count[p]=1;

pf++;

printf("Page Fault %d",pf);

}

for(j=0;j<f;j++)

printf("\t%d",fp[j]);

printf("\n");

}

printf("\n The number of Page Faults using LFu are %d",pf);

}