**Lab Assignment: 02**

**Title:** Implement different page replacement algorithms (FIFO, Optimal, LRU) using c.

**FIFO Theory:** This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

**Code:**

#include<stdio.h>

int main()

{

int i,j,n,a[50],frame[10],no,k,avail,count=0;

printf("ENTER THE NUMBER OF PAGES: ");

scanf("%d",&n);

printf("ENTER THE PAGE NUMBER: ");

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

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

printf("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]);

}

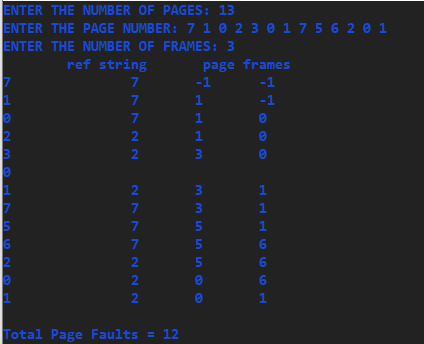
printf("\n");

}

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

return 0;

}

**Output:**

**Optimal:** The theoretically optimal page replacement algorithm (also known as OPT, clairvoyant replacement algorithm, or Bélády's optimal page replacement policy) is an algorithm that works as follows: when a page needs to be swapped in, the operating system swaps out the page whose next use will occur farthest in the future.

Code:

#include<stdio.h>

int main()

{

int f, n, frames[10], p[30], temp[10], flag1, flag2, flag3, i, j, k, pos, max, count = 0;

printf("ENTER THE NUMBER OF PAGES: ");

scanf("%d", &n);

printf("ENTER THE NUMBER OF FRAMES: ");

scanf("%d", &f);

printf("ENTER PAGE REFERENCE STRING: ");

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

{

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

}

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

{

frames[i] = -1;

}

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

{

flag1 = flag2 = 0;

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

{

if(frames[j] == p[i])

{

flag1 = flag2 = 1;

break;

}

}

if(flag1 == 0)

{

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

{

if(frames[j] == -1)

{

count++;

frames[j] = p[i];

flag2 = 1;

break;

}

}

}

if(flag2 == 0)

{

flag3 =0;

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

{

temp[j] = -1;

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

{

if(frames[j] == p[k])

{

temp[j] = k;

break;

}

}

}

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

{

if(temp[j] == -1)

{

pos = j;

flag3 = 1;

break;

}

}

if(flag3 ==0)

{

max = temp[0];

pos = 0;

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

{

if(temp[j] > max)

{

max = temp[j];

pos = j;

}

}

}

frames[pos] = p[i];

count++;

}

printf("\n");

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

{

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

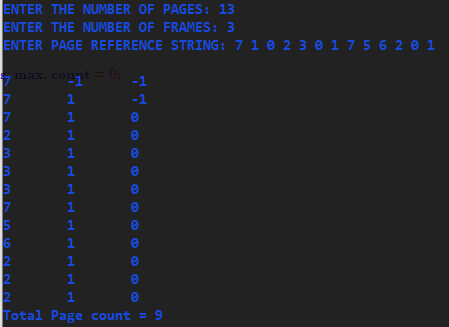
}

}

printf("\nTotal Page count = %d", count);

return 0;

}

**Output:**

**LRU:-**

Least Recently Used (LRU) algorithm is a page replacement technique used for memory management. According to this method, the page which is least recently used is replaced. Therefore, in memory, any page that has been unused for a longer period of time than the others is replaced.

**Code:**

#include<stdio.h>

int main()

{

int q[20],p[50],c=0,c1,d,f,i,j,k=0,n,r,t,b[20],c2[20];

printf("ENTER THE NUMBER OF PAGES: ");

scanf("%d",&n);

printf("Enter page reference string: ");

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

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

printf("ENTER THE NUMBER OF FRAMES :");

scanf("%d",&f);

q[k]=p[k];

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

c++;

k++;

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

{

c1=0;

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

{

if(p[i]!=q[j])

c1++;

}

if(c1==f)

{

c++;

if(k<f)

{

q[k]=p[i];

k++;

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

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

printf("\n");

}

else

{

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

{

c2[r]=0;

for(j=i-1; j<n; j--)

{

if(q[r]!=p[j])

c2[r]++;

else

break;

}

}

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

b[r]=c2[r];

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

{

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

{

if(b[r]<b[j])

{

t=b[r];

b[r]=b[j];

b[j]=t;

}

}

}

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

{

if(c2[r]==b[0])

q[r]=p[i];

printf("\t%d",q[r]);

}

printf("\n");

}

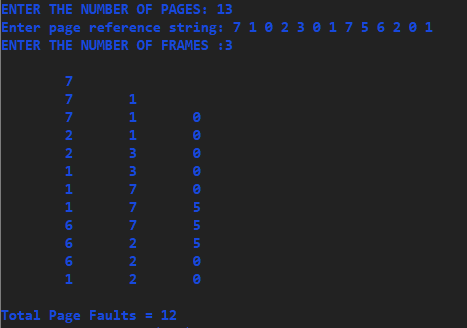
}

}

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

}

s

**Output:**