**8. Implement the following Page Replacement Algorithms**

**a) FIFO b) LFU c) LRU d) Optimal**

**a) First In First Out (FIFO)** **PAGE REPLACEMENT ALGORITHM**

This is the simplest page replacement algorithm. In this algorithm, the OS maintains a queue that keeps track of all the pages in memory, with the oldest page at the front and the most recent page at the back.

When there is a need for page replacement, the FIFO algorithm, swaps out the page at the front of the queue, that is the page which has been in the memory for the longest time.

**Note: Algorithm: students write description your own way…….**

**Program**

#include<stdio.h>

main()

{

int i, j, k, f, pf=0, count=0, rs[25], m[10], n;

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

scanf("%d",&n);

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

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

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

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

scanf("%d",&f);

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

m[i]=-1;

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

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

{

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

{

if(m[k]==rs[i])

break;

}

if(k==f)

{

m[count++]=rs[i];

pf++;

}

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

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

if(k==f)

printf("\tPF No. %d",pf);

printf("\n");

if(count==f)

count=0;

}

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

}

**Output:**

Enter the length of reference string -- 20

Enter the reference string -- 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

Enter no. of frames -- 3

The Page Replacement Process is --

7 -1 -1 PF No. 1

7 0 -1 PF No. 2

7 0 1 PF No. 3

2 0 1 PF No. 4

2 0 1

2 3 1 PF No. 5

2 3 0 PF No. 6

4 3 0 PF No. 7

4 2 0 PF No. 8

4 2 3 PF No. 9

0 2 3 PF No. 10

0 2 3

0 2 3

0 1 3 PF No. 11

0 1 2 PF No. 12

0 1 2

0 1 2

7 1 2 PF No. 13

7 0 2 PF No. 14

7 0 1 PF No. 15

**The number of Page Faults using FIFO are 15**

**b) LFU PAGE REPLACEMENT ALGORITHM**

The LFU page replacement algorithm stands for the **Least Frequently Used**. In the LFU page replacement algorithm, the page with the least visits in a given period of time is removed. It replaces the least frequently used pages. If the frequency of pages remains constant, the page that comes first is replaced first.

**Program**

#include<stdio.h>

main()

{

int rs[50], i, j, k, m, f, cntr[20], a[20], min, pf=0;

printf("\nEnter number of page references -- ");

scanf("%d",&m);

printf("\nEnter the reference string -- ");

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

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

printf("\nEnter the available no. of frames -- ");

scanf("%d",&f);

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

{

cntr[i]=0;

a[i]=-1;

}

printf("\nThe Page Replacement Process is – \n");

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

{

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

if(rs[i]==a[j])

{

cntr[j]++;

break;

}

if(j==f)

{

min = 0;

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

if(cntr[k]<cntr[min])

min=k;

a[min]=rs[i];

cntr[min]=1;

pf++;

}

printf("\n");

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

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

if(j==f)

printf(“\tPF No. %d”,pf);

}

printf("\n\n Total number of page faults -- %d",pf);

}

**Output:**

Enter number of page references -- 20

Enter the reference string – 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

Enter the available no. of frames -- 3

The Page Replacement Process is –

7 -1 -1 PF No. 1

7 0 -1 PF No. 2

7 0 1 PF No. 3

2 0 1 PF No. 4

2 0 1 PF No. 4

3 0 1 PF No. 5

3 0 1 PF No. 5

4 0 1 PF No. 6

2 0 1 PF No. 7

3 0 1 PF No. 8

3 0 1 PF No. 8

3 0 1 PF No. 8

3 0 2 PF No. 9

3 0 1 PF No. 10

3 0 2 PF No. 11

3 0 2 PF No. 11

3 0 1 PF No. 12

3 0 7 PF No. 13

3 0 7 PF No. 13

3 0 1 PF No. 14

Total number of page faults -- 14

**Output 2**

Enter number of page references -- 12

Enter the reference string -- 1

2

3

4

5

2

5

2

5

1

4

3

1

Enter the available no. of frames -- 3

The Page Replacement Process is –

1 -1 -1 PF No. 1

1 2 -1 PF No. 2

1 2 3 PF No. 3

4 2 3 PF No. 4

5 2 3 PF No. 5

5 2 3 PF No. 5

5 2 3 PF No. 5

5 2 3 PF No. 5

5 2 3 PF No. 5

5 2 1 PF No. 6

5 2 4 PF No. 7

5 2 3 PF No. 8

Total number of page faults – 8

**c) Least Recently Used (LRU)**

Least Recently Used page replacement algorithm keeps track of page usage over a short period of time. It works on the idea that the pages that have been most heavily used in the past are most likely to be used heavily in the future too.

In LRU, whenever page replacement happens, the page which has not been used for the longest amount of time is replaced.

#include<stdio.h>

main()

{

int i, j , k, min, rs[25], m[10], count[10], flag[25], n, f, pf=0, next=1;

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

scanf("%d",&n);

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

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

{

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

flag[i]=0;

}

printf("Enter the number of frames -- ");

scanf("%d",&f);

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

{

count[i]=0;

m[i]=-1;

}

printf("\nThe Page Replacement process is -- \n");

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

{

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

{

if(m[j]==rs[i])

{

flag[i]=1;

count[j]=next;

next++;

}

}

if(flag[i]==0)

{

if(i<f)

{

m[i]=rs[i];

count[i]=next;

next++;

}

else

{

min=0;

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

if(count[min] > count[j])

min=j;

m[min]=rs[i];

count[min]=next;

next++;

}

pf++;

}

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

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

if(flag[i]==0)

printf("PF No. -- %d" , pf);

printf("\n");

}

printf("\nThe number of page faults using LRU are %d",pf);

}

**OUTPUT:**

Enter the length of reference string -- 20

Enter the reference string -- 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

Enter the number of frames -- 3

The Page Replacement process is --

7 -1 -1 PF No. -- 1

7 0 -1 PF No. -- 2

7 0 1 PF No. -- 3

2 0 1 PF No. -- 4

2 0 1

2 0 3 PF No. -- 5

2 0 3

4 0 3 PF No. -- 6

4 0 2 PF No. -- 7

4 3 2 PF No. -- 8

0 3 2 PF No. -- 9

0 3 2

0 3 2

1 3 2 PF No. -- 10

1 3 2

1 0 2 PF No. -- 11

1 0 2

1 0 7 PF No. -- 12

1 0 7

1 0 7

**The number of page faults using LRU are 12**

**d) Optimal Page Replacement Algorithm**

Optimal Page Replacement algorithm is the **best page replacement** algorithm as it gives the **least number of page faults**. It is also known as **OPT**, clairvoyant replacement algorithm, or Belady’s optimal page replacement policy. In this algorithm, pages are replaced which would not be used for the longest duration of time in the future, i.e., the pages in the memory which are going to be referred farthest in the future are replaced.

This algorithm was introduced long back and is difficult to implement because it requires future knowledge of the program behaviour. However, it is possible to implement optimal page replacement on the second run by using the page reference information collected on the first run.