Page Replacement Algorithms

```
#include<stdio.h>
int n,nf;
int in[100];
int p[50];
int hit=0;
int i,j,k;
int pgfaultcnt=0;
 void getData()
{
    printf("\nEnter length of page 
reference sequence:");
    scanf("%d",&n);
    printf("\nEnter the page 
reference sequence:");
    for(i=0; i<n; i++)
        scanf("%d",&in[i]);
    printf("\nEnter no of 
frames:");
    scanf("%d",&nf);
}

void initialize()
{
    pgfaultcnt=0;
    for(i=0; i<nf; i++)
        p[i]=9999;
}

int isHit(int data)
{
    hit=0;
    for(j=0; j<nf; j++)
    {
        if(p[j]==data)
        {
            hit=1;
            break;
        }

    }

    return hit;
}

int getHitIndex(int data)
{
    int hitind;
    for(k=0; k<nf; k++)
    {
        if(p[k]==data)
        {
            hitind=k;
            break;
        }
    }
    return hitind;
}

void dispPages()
{
    for (k=0; k<nf; k++)
    {
        if(p[k]!=9999)
            printf(" %d",p[k]);
    }

}

void dispPgFaultCnt()
{
    printf("\nTotal no of page 
faults:%d",pgfaultcnt);
}

void fifo()
{
    initialize();
    for(i=0; i<n; i++)
    {
        printf("\nFor %d :",in[i]);

        if(isHit(in[i])==0)
        {

            for(k=0; k<nf-1; k++)
                p[k]=p[k+1];

            p[k]=in[i];
            pgfaultcnt++;
            dispPages();
        }
        else
            printf("No page 
fault");
    }
    dispPgFaultCnt();
}

void optimal()
{
    initialize();
    int near[50];
    for(i=0; i<n; i++)
    {

        printf("\nFor %d :",in[i]);

        if(isHit(in[i])==0)
        {
```

Page Replacement Algorithms

```
            for(j=0; j<nf; j++)
            {
                int pg=p[j];
                int found=0;
                for(k=i; k<n; k++)
                {
                    if(pg==in[k])
                    {
                        near[j]=k;
                        found=1;
                        break;
                    }
                    else
                        found=0;
                }
                if(!found)
                    near[j]=9999;
            }
            int max=-9999;
            int repindex;
            for(j=0; j<nf; j++)
            {
                if(near[j]>max)
                {
                    max=near[j];
                    repindex=j;
                }
            }
            p[repindex]=in[i];
            pgfaultcnt++;

            dispPages();
        }
        else
            printf("No page
fault");
    }
    dispPgFaultCnt();
}

void lru()
{
    initialize();

    int least[50];
    for(i=0; i<n; i++)
    {

        printf("\nFor %d :",in[i]);

        if(isHit(in[i])==0)
        {

            for(j=0; j<nf; j++)
            {
                int pg=p[j];
                int found=0;
                for(k=i-1; k>=0; k-
-)
                {
                    if(pg==in[k])
                    {
                        least[j]=k;
                        found=1;
                        break;
                    }
                    else
                        found=0;
                }
                if(!found)
                    least[j]=-9999;
            }
            int min=9999;
            int repindex;
            for(j=0; j<nf; j++)
            {
                if(least[j]<min)
                {
                    min=least[j];
                    repindex=j;
                }
            }
            p[repindex]=in[i];
            pgfaultcnt++;

            dispPages();
        }
        else
            printf("No page
fault!");
    }
    dispPgFaultCnt();
}

void lfu()
{
    int usedcnt[100];
    int least,repin,sofarcnt=0,bn;
    initialize();
    for(i=0; i<nf; i++)
        usedcnt[i]=0;

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

        printf("\n For %d
:",in[i]);
        if(isHit(in[i]))
        {
            int 
hitind=getHitIndex(in[i]);
            usedcnt[hitind]++;
            printf("No page 
fault!");
        }
        else
        {
            pgfaultcnt++;
            if(bn<nf)
            {
                p[bn]=in[i];
                usedcnt[bn]=usedcnt
[bn]+1;
                bn++;
            }
            else
            {
                least=9999;
                for(k=0; k<nf; k++)
                    if(usedcnt[k]<l
east)
                    {
                        least=usedc
nt[k];
                        repin=k;
                    }
                p[repin]=in[i];
                sofarcnt=0;
                for(k=0; k<=i; k++)
                    if(in[i]==in[k]
)
                        sofarcnt=so
farcnt+1;
                usedcnt[repin]=sofa
rcnt;
            }

            dispPages();
        }

    }
    dispPgFaultCnt();
}

void secondchance()
{
    int usedbit[50];
    int victimptr=0;
    initialize();
    for(i=0; i<nf; i++)
        usedbit[i]=0;
    for(i=0; i<n; i++)
    {
        printf("\nFor %d:",in[i]);
        if(isHit(in[i]))
        {
            printf("No page 
fault!");
            int 
hitindex=getHitIndex(in[i]);
            if(usedbit[hitindex]==0
)
                usedbit[hitindex]=1
;
        }
        else
        {
            pgfaultcnt++;
            if(usedbit[victimptr]==
1)
            {
                do
                {
                    usedbit[victimp
tr]=0;
                    victimptr++;
                    if(victimptr==n
f)
                        victimptr=0
;
                }
                while(usedbit[victi
mptr]!=0);
            }
            if(usedbit[victimptr]==
0)
            {
                p[victimptr]=in[i];
                usedbit[victimptr]=
1;
                victimptr++;
            }
            dispPages();

        }
        if(victimptr==nf)
            victimptr=0;
    }
    dispPgFaultCnt();
}

int main()
{
    int choice;
    while(1)
    {
        printf("\nPage Replacement 
Algorithms\n1.Enter 
data\n2.FIFO\n3.Optimal\n4.LRU\n5.L
FU\n6.Second Chance\n7.Exit\nEnter 
your choice:");
```

```
        scanf("%d",&choice);
        switch(choice)
        {
        case 1:
            getData();
            break;
        case 2:
            fifo();
            break;
        case 3:
            optimal();
            break;
        case 4:
            lru();
            break;
        case 5:
            lfu();
            break;
        case 6:
            secondchance();
            break;
        default:
            return 0;
            break;
        }
    }
}
```

OUTPUT:

```
Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
5.LFU
6.Second Chance
7.Exit
Enter your choice:1

Enter length of page reference sequ
ence:8

Enter the page reference sequence:2
3
4
2
3
5
6
2

Enter no of frames:3

Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
5.LFU
6.Second Chance
7.Exit
Enter your choice:2

For 2 : 2
For 3 : 2 3
For 4 : 2 3 4
For 2 :No page fault
For 3 :No page fault
For 5 : 3 4 5
For 6 : 4 5 6
For 2 : 5 6 2
Total no of page faults:6
Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
5.LFU
6.Second Chance
7.Exit
Enter your choice:3

For 2 : 2
For 3 : 2 3
For 4 : 2 3 4
For 2 :No page fault
For 3 :No page fault
For 5 : 2 5 4
For 6 : 2 6 4
For 2 :No page fault
Total no of page faults:5
Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
5.LFU
6.Second Chance
7.Exit
Enter your choice:4

For 2 : 2
For 3 : 2 3
For 4 : 2 3 4
For 2 :No page fault!
For 3 :No page fault!
For 5 : 2 3 5
For 6 : 6 3 5
For 2 : 6 2 5
Total no of page faults:6
Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
```

```
4.LRU
5.LFU
6.Second Chance
7.Exit
Enter your choice:5

 For 2 : 2
 For 3 : 2 3
 For 4 : 2 3 4
 For 2 :No page fault!
 For 3 :No page fault!
 For 5 : 2 3 5
 For 6 : 2 3 6
 For 2 :No page fault!
Total no of page faults:5
Page Replacement Algorithms
1.Enter data
2.FIFO
3.Optimal
4.LRU
5.LFU
6.Second Chance
7.Exit
Enter your choice:7
```