**AIM : 6.** Implement the C program for Page Replacement Algorithms: FCFS, LRU, and Optimal for frame size as minimum three.

```c
//LAB 6 – PRA.c

#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)
                {

                    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();
}



int main()
{
        int choice;
        while(1)
        {
            printf("\nPage Replacement Algorithms\n1.Enter
data\n2.FIFO\n3.Optimal\n4.LRU\n5.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:
                exit(0);
            default:
                return 0;
                break;
            }
        }
}
```