Code:

```
#include<iostream>
using namespace std;

int *initializeFrames(int frames[],int numOfFrames){
    for(int i=0;i<numOfFrames;i++){
        frames[i] = -1;
    }
    return frames;
}

//Supporting functions for LRU starts
int present(int table_frame[], int nf, int page)
{
    for(int i=0; i<nf; i++)
        if(page == table_frame[i])
            return 1;
    return 0;
}

void printtable(int table_frame[], int nf)
{
    for(int i=0; i<nf; i++)
    {
        if(table_frame[i] == -1)
            printf(" -\t");
        else
            printf("%2d\t", table_frame[i]);
    }
}

int findpos(int table_frame[], int nf, int pages[], int curr, int np)
{
    for(int i=0; i<nf; i++)
        if(table_frame[i] == -1)
            return i;

    int pos[nf] = {0};
    for(int i=0; i<nf; i++)
    {
        pos[i] = -1e9;
        for(int j=curr-1; j>=0; j--)
            if(pages[j] == table_frame[i])
            {
                pos[i] = j;
                break;
            }
    }

    int min1 = 1000000, retPos = -1;
    for(int i=0; i<nf; i++)
        if(min1 > pos[i])
        {
            min1 = pos[i];
            retPos = i;
        }
```

```
    return retPos;
}
//LRU Functions ENDS
void algoLRU(int numOfPages,int numOfFrames,int pageString[],int frames[]){
    bool pagePresentStatus;
    int pageFault, pageHit;
    frames = initializeFrames(frames,numOfFrames);
    /*for(int i=0;i<numOfPages;i++){
        for(int j=0;j<numOfFrames;j++)
        {
            if(pageString[i] == frames[j]) pagePresentStatus = true;
        }
        if(!pagePresentStatus){
            pageFault++;
        } else{
            pageHit++;
        }
        cout<<endl;
        for(int k=0;k<numOfFrames;k++){
            cout<<frames[k]<<" ";
        }
        pagePresentStatus = false;
    }
    cout<<"\n\nPage Faults : "<<pageFault<<endl;
    cout<<"Page Hits : "<<pageHit<<endl;*/
    int n=numOfPages,nf=numOfFrames,i,pos=0;
    pageFault=0;
    pageHit = 0;
    cout<<"\n**Page Replacement using Least Recently Used
Algorithm**\n"<<endl;
    for(i=0;i<n;i++)
    {
        //printf("Page %2d ->\t",pageString[i]);
        cout<<"Page "<<pageString[i]<<"\t->\t";
        if(!present(frames,nf,pageString[i]))
        {
            int pos = findpos(frames,nf,pageString,i,n);
            frames[pos]=pageString[i];

            printtable(frames,nf);
            printf("\'F\'\n");
            pageFault++;
            continue;
        }else{
            printtable(frames,nf);
            printf("\'H\'\n");
            pageHit++;
        }
    }
    printf("\nPage faults : %d", pageFault);
    printf("\nPage Hit : %d\n\n", pageHit);

}
void algoOptimal(int numOfPages,int numOfFrames,int pageString[],int
frames[]){
    int temp[10], flag1, flag2, flag3, i, j, k, pos, max,
```

```
pageFaults=0,pageHits=0;
    bool presentStatus;
    initializeFrames(frames,numOfFrames);
    cout<<endl<<"**Page Replacement using Optimal Algorithm**"<<endl;
    for(i = 0; i < numOfPages; ++i){
        flag1 = flag2 = 0;
        presentStatus = false;
        for(j = 0; j < numOfFrames; ++j){
            if(frames[j] == pageString[i]){
                flag1 = flag2 = 1;
                pageHits++;
                presentStatus = true;
                break;
            }
        }

        if(flag1 == 0){
            for(j = 0; j < numOfFrames; ++j){
                if(frames[j] == -1){
                    pageFaults++;
                    frames[j] = pageString[i];
                    flag2 = 1;
                    break;
                }
            }
        }

        if(flag2 == 0){
            flag3 =0;

            for(j = 0; j < numOfFrames; ++j){
                temp[j] = -1;

                for(k = i + 1; k < numOfPages; ++k){
                    if(frames[j] == pageString[k]){
                        temp[j] = k;
                        break;
                    }
                }
            }

            for(j = 0; j < numOfFrames; ++j){
                if(temp[j] == -1){
                    pos = j;
                    flag3 = 1;
                    break;
                }
            }

            if(flag3 ==0){
                max = temp[0];
                pos = 0;

                for(j = 1; j < numOfFrames; ++j){
                    if(temp[j] > max){
                        max = temp[j];
                        pos = j;
```

```
                    }
                }
            }

            frames[pos] = pageString[i];
            pageFaults++;
        }

        cout<<endl;
        cout<<"Page "<<pageString[i]<<"\t->\t";
        for(j = 0; j < numOfFrames; ++j){
            (frames[j]==-1)?cout<<"-"<<"\t":cout<<frames[j]<<"\t";
        }
        presentStatus?cout <<"\'H\'":cout <<"\'F\'";
    }

    cout<<"\n\nPage Faults : "<<pageFaults<<endl;
    cout<<"Page Hits : "<<pageHits<<endl<<endl;
}
void algoFIFO(int numOfPages,int numOfFrames,int pageString[],int frames[]){
    /*frames = initializeFrames(frames,numOfFrames);
    cout<<"num of pages : "<<numOfPages<<endl;
    cout<<"num of Frames : "<<numOfFrames<<endl;
    cout<<"String : ";
    for(int i =0;i<numOfPages;i++){
        cout<<" "<<pageString[i];
    }
    cout<<"\nFrames : ";
    for(int i =0;i<numOfFrames;i++){
        cout<<" "<<frames[i];
    }
    cout<<endl;*/
    int cachePosPointer = 0;
    bool presentStatus;
    int pageFault =0, pageHit = 0;
    frames = initializeFrames(frames,numOfFrames);
    cout<<endl<<"**Page Replacement using FIFO Algorithm**"<<endl;
    for(int i=0;i<numOfPages;i++){
        for(int j=0;j<numOfFrames;j++)
        {
            if(pageString[i] == frames[j]) presentStatus = true;
        }
        if(!presentStatus){
            if(cachePosPointer==numOfFrames) cachePosPointer = 0;
            frames[cachePosPointer++]=pageString[i];
            pageFault++;
        } else{
            pageHit++;
        }
        cout<<endl;
        cout<<"Page "<<pageString[i]<<"\t->\t";
        for(int k=0;k<numOfFrames;k++){
            frames[k]==-1?cout<<"-"<<"\t":cout<<frames[k]<<"\t";
        }
        presentStatus?cout <<"\'H\'":cout <<"\'F\'";
        presentStatus = false;
    }
```

```
    cout<<"\n\nPage Faults : "<<pageFault<<endl;
    cout<<"Page Hits : "<<pageHit<<endl<<endl;
}

int main(){
    int numOfPages,numOfFrames,choice;
    cout<<"Enter number of Pages : ";
    cin>>numOfPages;
    int pages[numOfPages];
    cout<<endl<<"Enter number of Frames : ";
    cin>>numOfFrames;
    int frames[numOfFrames];
    cout<<"\nEnter Pages ->\n";
    for(int r=0;r<numOfPages;r++){
        cout<<"Enter page "<<(r+1)<<" : ";
        cin>>pages[r];
    }
    while(1) {
        cout<<"|------------------------|\n";
        cout<<"| Menu                   |\n|------------------------|"
             "\n| 1: Least Recently Used |\n| 2: Optimal             |\n| 3:
First In First Out  |\n| 4: Exit                |\n|-----------------------
|\nEnter your choice : ";
                cin>>choice;
        switch (choice) {
            case 1:
                algoLRU(numOfPages,numOfFrames,pages,frames);
                break;
            case 2:
                algoOptimal(numOfPages,numOfFrames,pages,frames);
                break;
            case 3:
                algoFIFO(numOfPages,numOfFrames,pages,frames);
                break;
            case 4:
                cout<<endl<<"Terminated..!";
                exit(0);
                break;
            default:
                cout << "\nPlease Enter valid choice between 1 to 4\n";
        }
    }
    return 0;

}
```

Output:

```
Enter number of Pages :14

Enter number of Frames :4

Enter Pages ->
Enter page 1 :7
Enter page 2 :0
Enter page 3 :1
Enter page 4 :2
Enter page 5 :0
Enter page 6 :3
Enter page 7 :0
Enter page 8 :4
Enter page 9 :2
Enter page 10 :3
Enter page 11 :0
Enter page 12 :3
Enter page 13 :2
Enter page 14 :3
|--------------------------------|
| Menu                           |
|--------------------------------|
| 1: Least Recently Used         |
| 2: Optimal                     |
| 3: First In First Out          |
| 4: Exit                        |
|--------------------------------|
Enter your choice :1

**Page Replacement using Least Recently Used Algorithm**

Page 7  ->    7    -    -    -    'F'
Page 0  ->    7    0    -    -    'F'
Page 1  ->    7    0    1    -    'F'
Page 2  ->    7    0    1    2    'F'
Page 0  ->    7    0    1    2    'H'
Page 3  ->    3    0    1    2    'F'
Page 0  ->    3    0    1    2    'H'
Page 4  ->    3    0    4    2    'F'
Page 2  ->    3    0    4    2    'H'
Page 3  ->    3    0    4    2    'H'
Page 0  ->    3    0    4    2    'H'
Page 3  ->    3    0    4    2    'H'
```

```
Page 2  ->      3      0      4      2      'H'
Page 3  ->      3      0      4      2      'H'

Page faults : 6
Page Hit : 8

|---------------------------------|
| Menu                            |
|---------------------------------|
| 1: Least Recently Used          |
| 2: Optimal                      |
| 3: First In First Out           |
| 4: Exit                         |
|---------------------------------|
Enter your choice :2

**Page Replacement using Optimal Algorithm**

Page 7  ->      7      -      -      -      'F'
Page 0  ->      7      0      -      -      'F'
Page 1  ->      7      0      1      -      'F'
Page 2  ->      7      0      1      2      'F'
Page 0  ->      7      0      1      2      'H'
Page 3  ->      3      0      1      2      'F'
Page 0  ->      3      0      1      2      'H'
Page 4  ->      3      0      4      2      'F'
Page 2  ->      3      0      4      2      'H'
Page 3  ->      3      0      4      2      'H'
Page 0  ->      3      0      4      2      'H'
Page 3  ->      3      0      4      2      'H'
Page 2  ->      3      0      4      2      'H'
Page 3  ->      3      0      4      2      'H'

Page Faults : 6
Page Hits : 8

|---------------------------------|
| Menu                            |
|---------------------------------|
| 1: Least Recently Used          |
| 2: Optimal                      |
| 3: First In First Out           |
| 4: Exit                         |
|---------------------------------|
```

Enter your choice :3

Page Replacement using FIFO Algorithm

```
Page 7  ->    7     -     -     -    'F'
Page 0  ->    7     0     -     -    'F'
Page 1  ->    7     0     1     -    'F'
Page 2  ->    7     0     1     2    'F'
Page 0  ->    7     0     1     2    'H'
Page 3  ->    3     0     1     2    'F'
Page 0  ->    3     0     1     2    'H'
Page 4  ->    3     4     1     2    'F'
Page 2  ->    3     4     1     2    'H'
Page 3  ->    3     4     1     2    'H'
Page 0  ->    3     4     0     2    'F'
Page 3  ->    3     4     0     2    'H'
Page 2  ->    3     4     0     2    'H'
Page 3  ->    3     4     0     2    'H'
```

Page Faults : 7
Page Hits : 7

```
|---------------------------------|
| Menu                            |
|---------------------------------|
| 1: Least Recently Used          |
| 2: Optimal                      |
| 3: First In First Out           |
| 4: Exit                         |
|---------------------------------|
```

Enter your choice :4

Terminated..!
Process finished with exit code 0