Assignment – B7

Code

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

int refstring[] = {0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1};
int win, hit = 0, miss = 0, res[4], lastUsed[40];

int findLRU() {
    int minIndex = 0;
    for(int i = 1; i < win; i++){
        if(lastUsed[i] < lastUsed[minIndex]){
            minIndex = i;
        }
    }
    return minIndex;
}

void lru(int rf[], int len)
{
    for(int i = 0; i < win; i++) {
        res[i] = -1;
        lastUsed[i] = -1;
    }
    for(int j = 0; j < len; j++) {
        bool hitFlag = false;
        for(int k = 0; k < win; k++){
            if (res[k] == rf[j]){
                hit++;
                lastUsed[k] = j;
                hitFlag = true;
                break;
            }
        }
        if (!hitFlag) {
            miss++;
            bool emptyFound = false;
            for(int k = 0; k < win; k++){
                if (res[k] == -1){
                    res[k] = rf[j];
                    lastUsed[k] = j;
                    emptyFound = true;
                   break;
                }
            }
            if(!emptyFound){
                int lruIndex = findLRU();
                res[lruIndex] = rf[j];
                lastUsed[lruIndex] = j;
            }
```

```
        }
    }

    cout<<"\nNo of hits (LRU): "<<hit;
    cout<<"\nNo of misses (LRU): "<<miss;
}
int findOptimal(int rf[], int len, int currentIndex)
{
    int maxIndex = -1, farthest = currentIndex;
    for(int i = 0; i < win; i++){
        int j;
        for(j = currentIndex + 1; j < len; j++){
            if(res[i] == rf[j]){
                if(j > farthest){
                    farthest = j;
                    maxIndex = i;
                }
                break;
            }
        }
        if(j == len)
            return i;
    }
    return maxIndex;
}

void optimal(int rf[], int len){
    hit = 0;
    miss = 0;

    for(int i = 0; i < win; i++){
        res[i] = -1;
    }

    for(int j = 0; j < len; j++){
        bool hitFlag = false;
        for(int k = 0; k < win; k++){
            if (res[k] == rf[j]){
                hit++;
                hitFlag = true;
                break;
            }
        }
        if (!hitFlag){
           miss++;
            bool emptyFound = false;
            for(int k = 0; k < win; k++){
                if (res[k] == -1){
                    res[k] = rf[j];
                    emptyFound = true;
                    break;
                }
            }
```

```
            if(!emptyFound){
                int optimalIndex = findOptimal(rf, len, j);
                res[optimalIndex] = rf[j];
            }
        }
    }

    cout<<"\nNo of hits (Optimal): "<<hit;
    cout<<"\nNo of misses (Optimal): "<<miss;
}

void display()
{
    cout<<"\nPages in window are: \n";
    for(int i = 0; i < win; i++){
        cout<<res[i]<<"\n";
    }
}

int main()
{
    cout<<"\nEnter window size: ";
    cin>>win;
    int len = sizeof(refstring)/sizeof(refstring[0]);

    cout<<"\nBy LRU algorithm: ";
    lru(refstring, len);
    display();

    cout<<"\n\nBy Optimal Algorithm: ";
    optimal(refstring, len);
    display();
}
```

Output

```
$ g++ Code-B7.cpp && ./a.out

Enter window size: 3

By LRU algorithm:
No of hits (LRU): 3
No of misses (LRU): 9
Pages in window are:
1
3
2


By Optimal Algorithm:
No of hits (Optimal): 5
No of misses (Optimal): 7
Pages in window are:
2
4
1
```