Caching Simulation

```
#include <bits/stdc++.h>
#include <windows.h>
#include <conio.h>
#define n 8
#define symbol '_'
#define WINDOWS 1

using namespace std;

void clrscr() {
  #ifdef WINDOWS
  system("cls");
  #endif
  #ifdef LINUX
  system("clear");
  #endif
}

// variables
bool flag;
int no_of_page_ref;
int *page_ref_array;
int hit = 0,miss = 0;
int cache_memory[n+1][n+1];

// functions
void read_cache(){
    ifstream fin;
    fin.open("Cache Memory.txt");

    for(int i=1; i<=n; i++)
    for(int j=1; j<=n; j++){

    int data;
    fin.read((char*)&data,sizeof(int));
    cache_memory[i][j]=data;
    }


    fin.close();

}

void write_cache(){
    ofstream fout;
    fout.open("Cache Memory.txt");
    int data;
    for(int i=1; i<=n; i++){
    for(int j=1; j<=n; j++){
    data = cache_memory[i][j];
    fout.write((char*)&data,sizeof(int));
    }
    }

    fout.close();
}

void print_cache(){
        cout<<"Cache Memory:\n";
        for(int i = 1;i<=n;i++)
        {
            for(int j = 1;j<=n;j++ )
            {
                if(cache_memory[i][j] == -1)
                cout<<setw(5)<<symbol<<" ";
                else
               cout<<setw(5)<<cache_memory[i][j]<<" ";
            }
            cout<<endl;
        }
}

void read_input(){
cout<<"How many pages you want?"<<endl;
                                //
    cin>>no_of_page_ref;
    if(no_of_page_ref <= 0 ) return;
    page_ref_array = new int[no_of_page_ref];
    cout<<"Enter the page references you want:"<<endl;


    for(int i = 1; i <= no_of_page_ref; i++)
    {
        cin>>page_ref_array[i];
    }
}


void page_referencing(){
    int o = no_of_page_ref;
    int a = 1;
      cout<<"Hit count: "<<hit<<endl;
      cout<<"Miss count: "<<miss<<endl;
    while(o--)
    {
        int temp = page_ref_array[a];

        // cache hit  case:
        flag = 0;
        for(int i = 1; i<=n; i++)
        {
            for(int j = 1; j<=n; j++){
                if(cache_memory[i][j] == temp)
                {
                    cout<<" STATUS: cache hit"<<endl;
                    hit++;
                    flag = 1;
```

```
                        goto state1;
                    }
                }
            }

            state1:
            if(flag == 1)
            {
                a++;
                print_cache();
              clrscr();
              cout<<"How many pages you want?"<<endl;
              cout<<no_of_page_ref<<endl;
              cout<<"Enter the page references you want:"<<endl;
              for(int i =1;i<=no_of_page_ref;i++)
             {
                 cout<<page_ref_array[i]<<" ";
             }
             cout<<endl;
             cout<<"Current hit: "<<hit<<endl;
             cout<<"Current miss: "<<miss<<endl;
                continue;
            }


            // cache miss  case:

            // If matrix is incompletely filled
            cout<<"STATUS: cache miss "<<endl;
            miss++;
            flag  = 0;
            for(int i = 1; i<=n; i++)
            {
                for(int j = 1; j<=n; j++)
                {
                    if(cache_memory[i][j] == -1)
                    {
                        flag = 1;
                        cache_memory[i][j] = page_ref_array[a];
                        goto state1;
                    }

                    }
            }

            //state2:
            // if matrix is completely filled and we have to replace

            int index = -1;
            flag= 0;
            int idx_max = INT_MIN;                              //index of
farthest incoming page reference
            pair<int,int> idx_of_cacheMem;                      //index of
cache Memory which is going to be replaced
```

```
    for(int i =1; i<=n; i++)
    {
        for(int j=1; j<=n ;j++)
        {
            flag= 0;
            int temp = cache_memory[i][j];

            //traversing the page reference array
            for(int k=a+1; k<=no_of_page_ref; k++)
            {
                if(temp == page_ref_array[k])
                {
                    flag = 1;
                    index = k;
                    break;
                }
            }

            // If required page will not come in future
            if(flag == 0)
            {
                cache_memory[i][j] = page_ref_array[a];
                goto statement;
            }

            // find farthest data
            else
            {
                if(index > idx_max )
                {
                    idx_max = index;
                    idx_of_cacheMem.first = i;
                    idx_of_cacheMem.second = j;
                }
                  }

        }
    }


    // Replacing farthest data

    int x,y;
    x = idx_of_cacheMem.first;
    y = idx_of_cacheMem.second;
    cache_memory[x][y] = temp;
    statement:
    a++;

    print_cache();

    Sleep(2000);
  clrscr();
  cout<<"How many pages you want?"<<endl;
  cout<<no_of_page_ref<<endl;
```

```
        cout<<"Enter the page references you want:"<<endl;
        cout<<"*****\n";
           for(int i =0;i<no_of_page_ref;i++)
           {
               cout<<page_ref_array[i]<<" ";
           }
           cout<<endl;
         cout<<"*****\n";
    }
}

int main() {
    read_cache();

    do{
    read_input();

    page_referencing();
    }while(no_of_page_ref > 0);

    write_cache();

   cout<<"Total hit is: "<<hit<<endl;
   cout<<"Total miss is: "<<miss<<endl;

    return 0;
}
```