###### Experiment Number: 05

###### TITLE: Page Replacement Algorithms

# First In First Out

## Code

*#include* <iostream>

*#include* <stdio.h>

using namespace std;

*// only give positive nos. as input*

int main()

{

    int arr[5], n, i, l, page[40];

    l = 4; *// no of the pages*

    cout << "Enter no  of inputs to be given:";

    cin >> n;

    cout << "Enter the nos to be taken as input:";

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

    {

        cin >> page[i];

    }

*for* (i = 0; i < l; i++)

    {

        arr[i] = -1;

    }

    int end\_index;

    int j = 0;

    int count = 0, flag = 0;

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

    {

        int hit = 0;

*for* (j = 0; j < l; j++)

        {

*if* (page[i] == arr[j])

            {

                cout << "\nPage hit";

                hit = 1;

*break*;

            }

*else* *if* (arr[j] == -1)

                end\_index = j;

*else*

                end\_index = l + 1;

        }

*if* (hit == 0 && end\_index <= l - 1) *// only for the first element*

        {

            cout << "\nPage Miss memory not full";

            count = count + 1;

            arr[end\_index] = page[i];

        }

*else* *if* (hit == 0 && end\_index == l + 1) *// memory full*

        {

            cout << "\nPage Miss memory full";

            count = count + 1;

*for* (j = 0; j < l - 1; j++)

            {

                arr[j] = arr[j + 1];

            }

            arr[j] = page[i];

        }

        cout << "\n";

*for* (int k = 0; k < l; k++)

        {

            cout << arr[k] << " ";

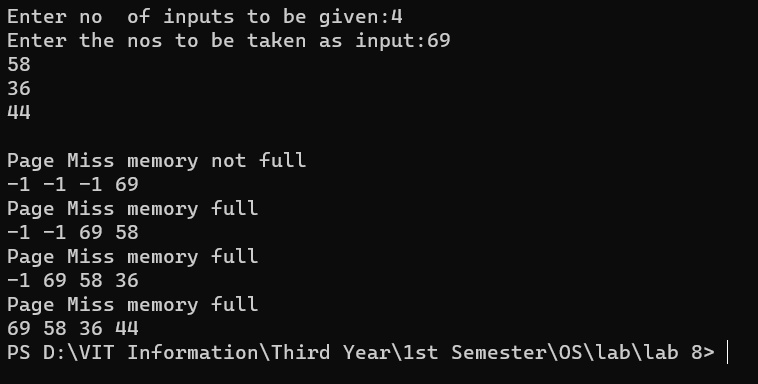
        }

    }

*return* 0;

}

## Output



# Least Recently Used

## Code

*#include* <iostream>

*#include* <stdio.h>

using namespace std;

int main()

{

    int i, n, page[50], arr[5], k, s, j, large, large\_index, dist[5], hit = 0, found = 0, f;

    int nhit = 0, nmiss = 0;

    int m = 0;

    cout << "\nEnter the no of pages to be filled in the page table:";

    cin >> n;

    cout << "\nEnter the no. of frames:";

    cin >> f;

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

        cin >> page[i];

    int finish = 0; *// keeps track of the last frame that was inserted into the page table*

    int pointer = 0; *// keeps track of the last location filled in the page table*

*// initializing the page table with -1*

*for* (i = 0; i < f; i++)

        arr[i] = -1;

*// filling the page table*

*for* (i = 0; i < n; i++) *// iterating through the frames in physical memory*

    {

*for* (j = 0; j <= i; j++) *// iterating through page table to check if the input frame is already present in the page table or not*

        {

*if* (page[i] != arr[j])

                hit = 0;

*else*

            {

                hit = 1;

                nhit = nhit + 1;

                cout << "\nPage Hit!";

*break*;

            }

        }

*if* (hit == 0) *// if not presnt i.e. page miss ,then inserted into the page table*

        {

            cout << "\n j is:" << j;

            nmiss = nmiss + 1;

            cout << "\nPage Miss!";

            arr[pointer] = page[i];

            pointer = pointer + 1;

        }

*// Displaying the current state of the array!*

        cout << "\nArray:";

*for* (k = 0; k < f; k++)

            cout << arr[k] << " ";

        cout << "\n";

*// checking if all locations of the page table have been filled*

*if* (pointer == f)

        {

            finish = i + 1;

            cout << "\nCompleted filling the frames";

*break*;

        }

    }

    hit = 0;

    found = 0;

*// optimal page replacement*

*for* (i = finish; i < n; i++) *// traversing the frames*

    {

*for* (j = 0; j < f;) *// traversing the page table*

        {

*if* (page[i] != arr[j])

            {

                j = j + 1;

                hit = 0;

            }

*else*

            {

                hit = 1;

*break*;

            }

        }

*if* (hit == 1)

        {

*// do nothing*

            cout << "\nPage hit!";

            nhit = nhit + 1;

        }

*else*

        {

*// Page fault*

            cout << "\nPage miss!";

            nmiss = nmiss + 1;

            int index = i;

*for* (k = 0; k < f; k++)

            {

                cout << "\nindex:" << index;

*// checking each element of the page table with the remaing array of frames, to check*

*// for the one with the longest forward distance*

*for* (m = index; m > 0; m--)

                {

*if* (arr[k] == page[m])

                    {

                        dist[k] = index - m;

                        cout << "\n m is " << m;

                        cout << "\n"

                             << arr[k] << "alloted a distance " << int(dist[k]);

                        found = 1;

*break*;

                    }

                }

*if* (found == 0) *// not found in frame array ,assigned a very large value*

                {

                    dist[k] = 99;

                    cout << "\n"

                         << arr[k] << "\nalloted a distance 99";

                }

            }

*// finding the one with largest backward distance and then replacing it*

            large = dist[0];

            large\_index = 0;

*for* (s = 1; s < f; s++)

            {

*if* (large < dist[s])

                {

                    large = dist[s];

                    large\_index = s;

                }

            }

            cout << "\n"

                 << arr[large\_index] << " stands with largest distance";

            arr[large\_index] = page[i];

        }

        cout << "\n";

*for* (k = 0; k < f; k++)

            cout << arr[k] << " ";

    }

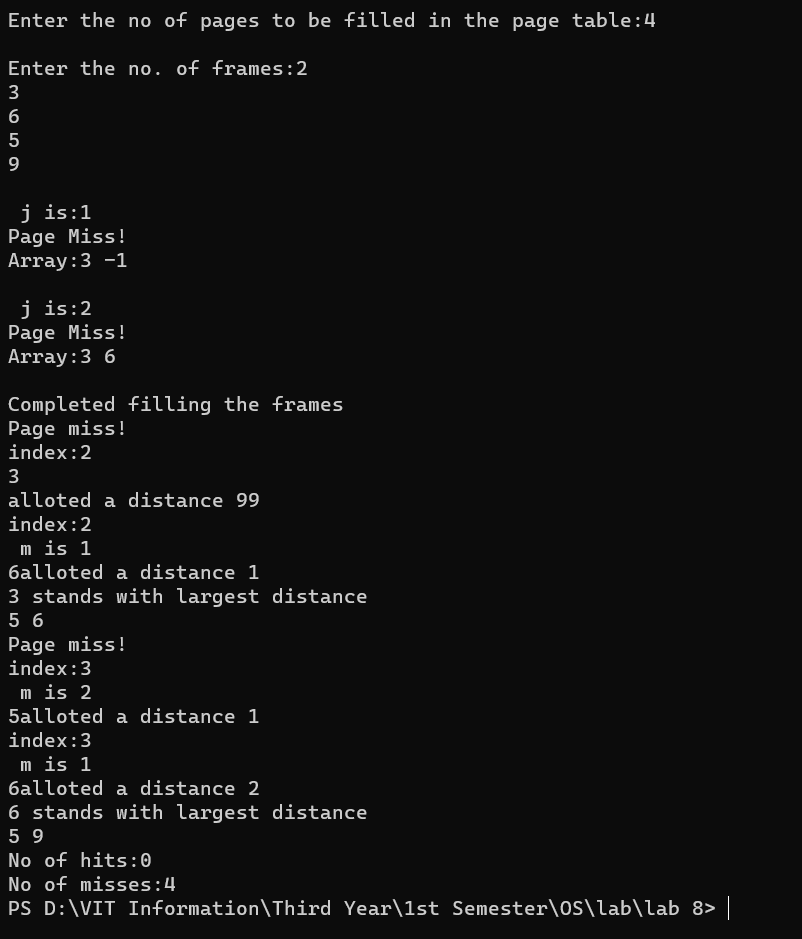
    cout << "\nNo of hits:" << nhit;

    cout << "\nNo of misses:" << nmiss;

*return* 0;

}

## Output



# Optimal

## Code

*#include* <iostream>

*#include* <stdio.h>

using namespace std;

int main()

{

    int i, n, page[50], arr[5], k, s, j, large, large\_index, dist[5], hit = 0, found = 0, f;

    int nhit = 0, nmiss = 0;

    int m = 0;

    cout << "\nEnter the no of pages to be filled in the page table:";

    cin >> n;

    cout << "\nEnter the no. of frames:";

    cin >> f;

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

        cin >> page[i];

    int finish = 0; *// keeps track of the last frame that was inserted into the page table*

    int pointer = 0; *// keeps track of the last location filled in the page table*

*// initializing the page table with -1*

*for* (i = 0; i < f; i++)

        arr[i] = -1;

*// filling the page table*

*for* (i = 0; i < n; i++) *// iterating through the frames in physical memory*

    {

*for* (j = 0; j <= i; j++) *// iterating through page table to check if the input frame is already present in the page table or not*

        {

*if* (page[i] != arr[j])

                hit = 0;

*else*

            {

                hit = 1;

                nhit = nhit + 1;

                cout << "\nPage Hit!";

*break*;

            }

        }

*if* (hit == 0) *// if not presnt i.e. page miss ,then inserted into the page table*

        {

            cout << "\n j is:" << j;

            nmiss = nmiss + 1;

            cout << "\nPage Miss!";

            arr[pointer] = page[i];

            pointer = pointer + 1;

        }

*// Displaying the current state of the array!*

        cout << "\nArray:";

*for* (k = 0; k < f; k++)

            cout << arr[k] << " ";

        cout << "\n";

*// checking if all locations of the page table have been filled*

*if* (pointer == f)

        {

            finish = i + 1;

            cout << "\nCompleted filling the frames";

*break*;

        }

    }

    hit = 0;

    found = 0;

*// optimal page replacement*

*for* (i = finish; i < n; i++) *// traversing the frames*

    {

*for* (j = 0; j < f;) *// traversing the page table*

        {

*if* (page[i] != arr[j])

            {

                j = j + 1;

                hit = 0;

            }

*else*

            {

                hit = 1;

*break*;

            }

        }

*if* (hit == 1)

        {

*// do nothing*

            cout << "\nPage hit!";

            nhit = nhit + 1;

        }

*else*

        {

*// Page fault*

            cout << "\nPage miss!";

            nmiss = nmiss + 1;

*for* (k = 0; k < f; k++)

            {

                int index = i;

                cout << "\nindex:" << index;

*// checking each element of the page table with the remaing array of frames, to check*

*// for the one with the longest forward distance*

*for* (m = index; m < n; m++)

                {

*if* (arr[k] == page[m])

                    {

                        dist[k] = m - index;

                        cout << "\n m is " << m;

                        cout << "\n"

                             << arr[k] << "alloted a distance " << int(dist[k]);

                        found = 1;

*break*;

                    }

                }

*if* (found == 0) *// not found in frame array ,assigned a very large value*

                {

                    dist[k] = 99;

                    cout << "\n"

                         << arr[k] << "\nalloted a distance 99";

                }

            }

*// finding the one with largest forward distance and then replacing it*

            large = dist[0];

            large\_index = 0;

*for* (s = 1; s < f; s++)

            {

*if* (large < dist[s])

                {

                    large = dist[s];

                    large\_index = s;

                }

            }

            cout << "\n"

                 << arr[large\_index] << " stands with largest distance";

            arr[large\_index] = page[i];

        }

        cout << "\n";

*for* (k = 0; k < f; k++)

            cout << arr[k] << " ";

    }

    cout << "\nNo of hits:" << nhit;

    cout << "\nNo of misses:" << nmiss;

*return* 0;

}

## Output

