MADAN BHANDARI MEMORIAL COLLEGE

New Baneshwor, Kathmandu

School of Science and Technology

Lab Report of Operating System

Submitted by:

Sarfraj Alam

Bsc.CSIT(4th Sem)

Section-'A'

Submitted to:

Sukraj Neyong

Lab no: 1 **Date: 2082/04/27**

Title: Write a program to display the number of page fault for user input reference string and frame size using FIFO.

Definition:

FIFO stands for First-In, First-Out. It's a basic page replacement algorithm used in operating systems to manage memory.

How FIFO Works:

- Pages are loaded into memory one by one.
- If a page is already in memory → Page Hit (no replacement needed).
- If a page is not in memory → Page Fault .
- If there's space → just load it.
- If memory is full → remove the oldest page (the one that came in earliest).
- Insert the new page.

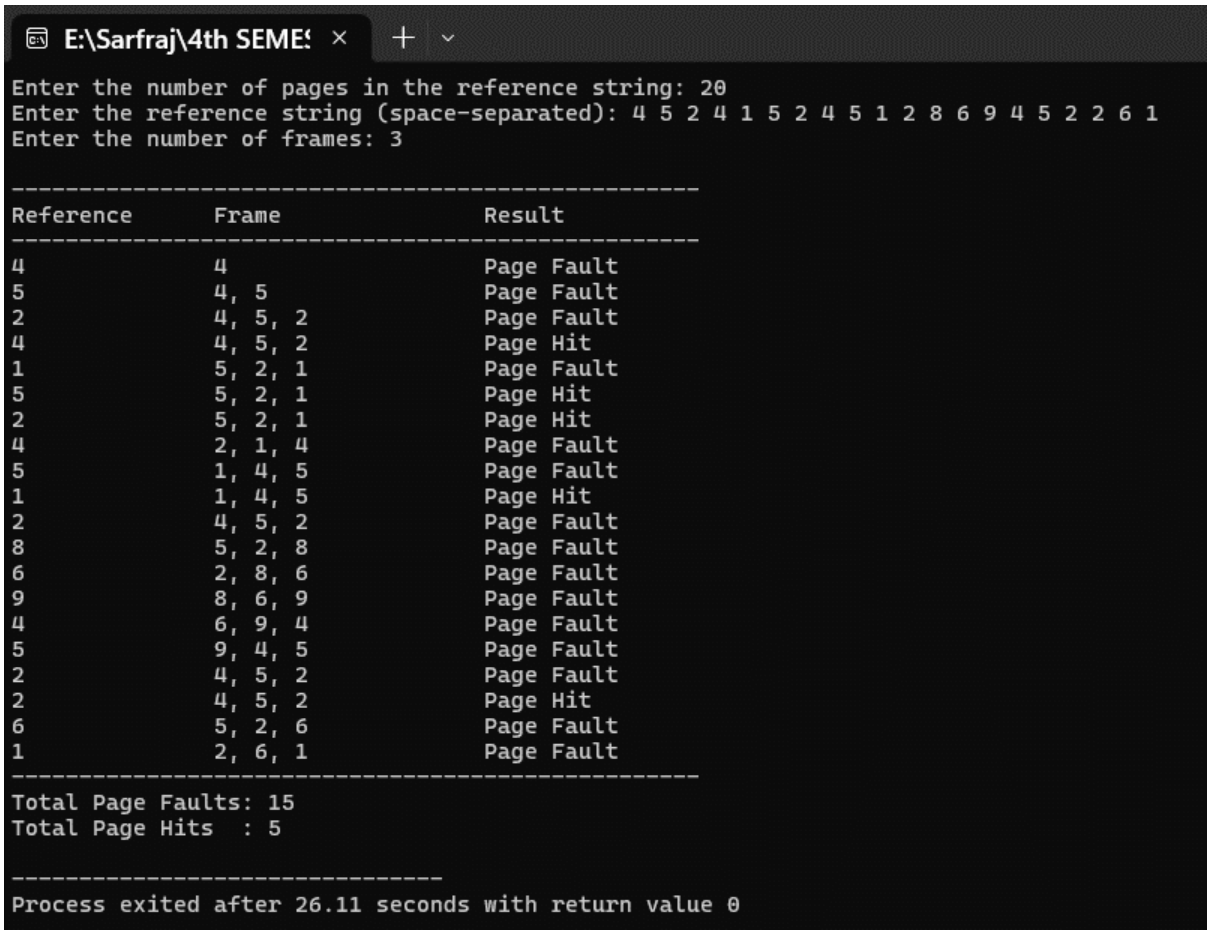
IDE : DEV C++

Source code:

```
#include <iostream>
#include <vector>
#include <queue>
#include <algorithm>
#include <iomanip>
#include <sstream>  // include stringstream header

using namespace std;

// Function to convert int to string for older compilers
string intToString(int num) {
    stringstream ss;
    ss << num;
    return ss.str();
}

int main() {
    int n, frameSize;
    vector<int> referenceString;
    queue<int> fifoQueue;
    vector<int> frameVec;
    int pageFaults = 0;

    // Input
    cout << "Enter the number of pages in the reference string: ";
    cin >> n;

    cout << "Enter the reference string (space-separated): ";
    for (int i = 0; i < n; ++i) {
        int page;
        cin >> page;
        referenceString.push_back(page);
    }

    cout << "Enter the number of frames: ";
    cin >> frameSize;

    // Header
    cout << "\n-----------------------------------------------------\n";
    cout << left << setw(15) << "Reference"
         << setw(20) << "Frame"
         << setw(15) << "Result" << endl;
    cout << "-----------------------------------------------------\n";

    // FIFO Logic with Output
    for (int i = 0; i < n; ++i) {
        int currentPage = referenceString[i];
        string result;

        vector<int>::iterator it = find(frameVec.begin(), frameVec.end(), currentPage);

        if (it != frameVec.end()) {
            // Page Hit
            result = "Page Hit";
```

```
        } else {
            // Page Fault
            result = "Page Fault";
            pageFaults++;

            if ((int)frameVec.size() < frameSize) {
                fifoQueue.push(currentPage);
                frameVec.push_back(currentPage);
            } else {
                int toRemove = fifoQueue.front();
                fifoQueue.pop();

                vector<int>::iterator it = find(frameVec.begin(), frameVec.end(), toRemove);
                if (it != frameVec.end()) frameVec.erase(it);

                fifoQueue.push(currentPage);
                frameVec.push_back(currentPage);
            }
        }

        // Display output for this step
        cout << left << setw(15) << currentPage;

        // Frame contents string
        string frameContent;
        for (size_t j = 0; j < frameVec.size(); ++j) {
            frameContent += intToString(frameVec[j]);  // use intToString here
            if (j != frameVec.size() - 1) frameContent += ", ";
        }

        cout << setw(20) << frameContent
             << setw(15) << result << endl;
    }

    // Final Summary
    cout << "--------------------------------------------------\n";
    cout << "Total Page Faults: " << pageFaults << endl;
    cout << "Total Page Hits  : " << n - pageFaults << endl;

    return 0;
}
```

Output:


E:\Sarfraj\4th SEME
Enter the number of pages in the reference string: 20
Enter the reference string (space-separated): 4 5 2 4 1 5 2 4 5 1 2 8 6 9 4 5 2 2 6 1
Enter the number of frames: 3
Reference Frame Result
4 4 Page Fault
5 4, 5 Page Fault
2 4, 5, 2 Page Fault
4 4, 5, 2 Page Hit
1 5, 2, 1 Page Fault
5 5, 2, 1 Page Hit
2 5, 2, 1 Page Hit
4 2, 1, 4 Page Fault
5 1, 4, 5 Page Fault
1 1, 4, 5 Page Hit
2 4, 5, 2 Page Fault
8 5, 2, 8 Page Fault
6 2, 8, 6 Page Fault
9 8, 6, 9 Page Fault
4 6, 9, 4 Page Fault
5 9, 4, 5 Page Fault
2 4, 5, 2 Page Fault
2 4, 5, 2 Page Hit
6 5, 2, 6 Page Fault
1 2, 6, 1 Page Fault
Total Page Faults: 15
Total Page Hits : 5
Process exited after 26.11 seconds with return value 0

Lab no: 2 **Date: 2082/04/27**

Title: Write a program to display the number of page fault for user input reference string and frame size using OPR.

Definition:

OPR stands for Optimal Page Replacement. It's a theoretical algorithm that always makes the best possible decision to minimize page faults.

How OPR Works:

For each page reference:

- If the page is already in memory → Page Hit
- If not → Page Fault

If memory is full:

- Look ahead in the reference string.
- Find the page in memory that will be used farthest in the future (or not at all).
- Replace that page with the new one.

IDE : DEV C++

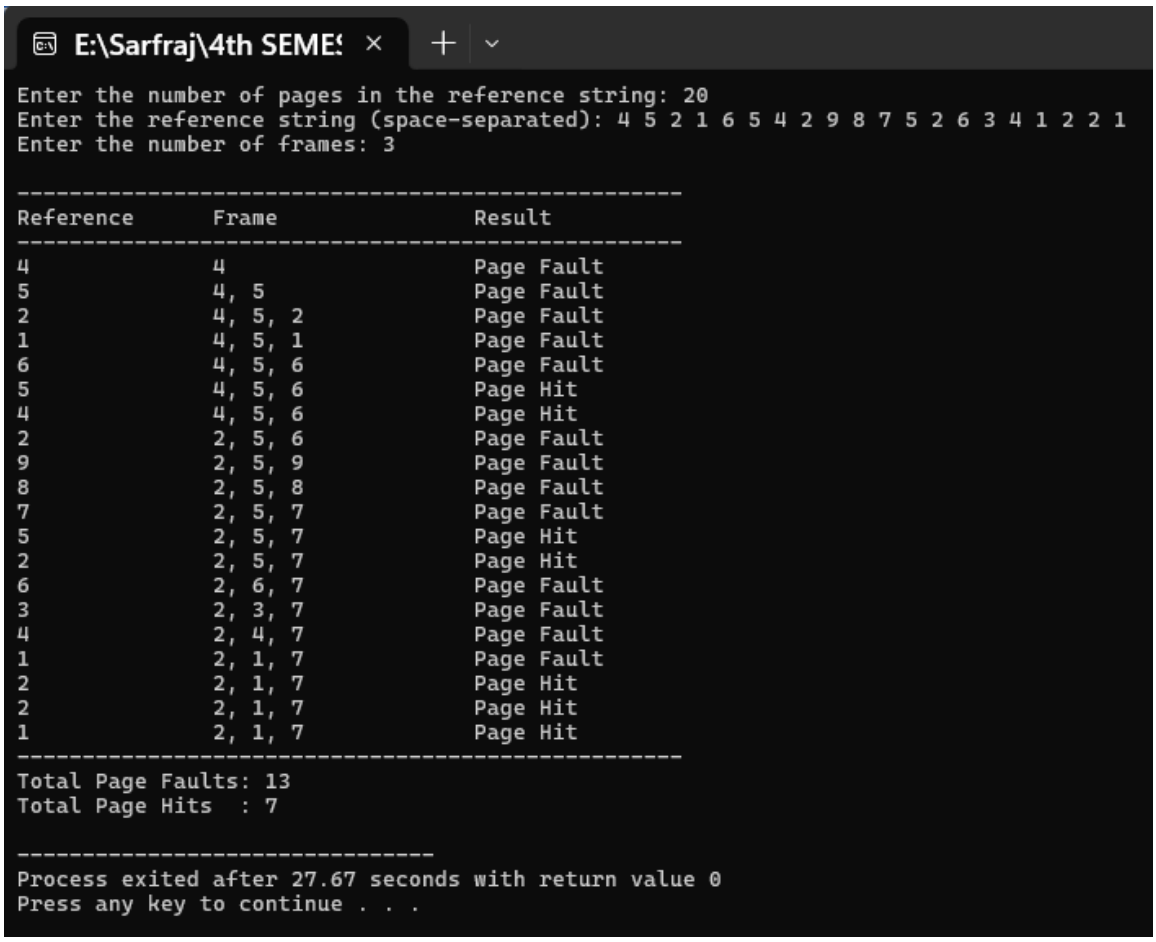
Source code:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <iomanip>
#include <sstream>

using namespace std;

// Convert int to string for older compilers
string intToString(int num) {
    stringstream ss;
    ss << num;
    return ss.str();
}

// Function to find the index of the page to replace using Optimal algorithm
int findOptimalIndex(const vector<int>& frameVec, const vector<int>& referenceString, int
currentIndex) {
    int farthest = currentIndex;
    int indexToReplace = -1;

    for (size_t i = 0; i < frameVec.size(); ++i) {
        int j;
        // Look ahead in the reference string to find when this page will be used next
        for (j = currentIndex; j < (int)referenceString.size(); ++j) {
            if (frameVec[i] == referenceString[j]) {
                if (j > farthest) {
                    farthest = j;
                    indexToReplace = (int)i;
                }
                break;
            }
        }
        // If page not found in future, replace this one immediately
        if (j == (int)referenceString.size()) {
            return i;
        }
    }

    // If all pages are found in future, replace the one with farthest use
    if (indexToReplace == -1) {
        // No page found farthest (means all pages used immediately), replace first frame by
default
        return 0;
    } else {
        return indexToReplace;
    }
}
```

```
int main() {
    int n, frameSize;
    vector<int> referenceString;
    vector<int> frameVec;
    int pageFaults = 0;

    // Input
    cout << "Enter the number of pages in the reference string: ";
    cin >> n;

    cout << "Enter the reference string (space-separated): ";
    for (int i = 0; i < n; ++i) {
        int page;
        cin >> page;
        referenceString.push_back(page);
    }

    cout << "Enter the number of frames: ";
    cin >> frameSize;

    // Header
    cout << "\n-----------------------------------------------------\n";
    cout << left << setw(15) << "Reference"
         << setw(20) << "Frame"
         << setw(15) << "Result" << endl;
    cout << "-----------------------------------------------------\n";

    for (int i = 0; i < n; ++i) {
        int currentPage = referenceString[i];
        string result;

        // Check if page is already in frames (Page Hit)
        vector<int>::iterator it = find(frameVec.begin(), frameVec.end(), currentPage);

        if (it != frameVec.end()) {
            result = "Page Hit";
        } else {
            // Page Fault
            result = "Page Fault";
            pageFaults++;

            if ((int)frameVec.size() < frameSize) {
                // Still have room in frames, just add page
                frameVec.push_back(currentPage);
            } else {
                // Need to replace a page optimally
                int indexToReplace = findOptimalIndex(frameVec, referenceString, i + 1);
                frameVec[indexToReplace] = currentPage;
            }
        }
```

```
        // Display output for this step
        cout << left << setw(15) << currentPage;

        // Frame contents string
        string frameContent;
        for (size_t j = 0; j < frameVec.size(); ++j) {
            frameContent += intToString(frameVec[j]);
            if (j != frameVec.size() - 1) frameContent += ", ";
        }

        cout << setw(20) << frameContent
             << setw(15) << result << endl;
    }

    // Final Summary
    cout << "-------------------------------------------------\n";
    cout << "Total Page Faults: " << pageFaults << endl;
    cout << "Total Page Hits  : " << n - pageFaults << endl;

    return 0;
}
```

Output:


E:\Sarfraj\4th SEME
Enter the number of pages in the reference string: 20
Enter the reference string (space-separated): 4 5 2 1 6 5 4 2 9 8 7 5 2 6 3 4 1 2 2 1
Enter the number of frames: 3
Reference Frame Result
4 4 Page Fault
5 4, 5 Page Fault
2 4, 5, 2 Page Fault
1 4, 5, 1 Page Fault
6 4, 5, 6 Page Fault
5 4, 5, 6 Page Hit
4 4, 5, 6 Page Hit
2 2, 5, 6 Page Fault
9 2, 5, 9 Page Fault
8 2, 5, 8 Page Fault
7 2, 5, 7 Page Fault
5 2, 5, 7 Page Hit
2 2, 5, 7 Page Hit
6 2, 6, 7 Page Fault
3 2, 3, 7 Page Fault
4 2, 4, 7 Page Fault
1 2, 1, 7 Page Fault
2 2, 1, 7 Page Hit
2 2, 1, 7 Page Hit
1 2, 1, 7 Page Hit
Total Page Faults: 13
Total Page Hits : 7
Process exited after 27.67 seconds with return value 0
Press any key to continue . . .

Lab no: 3 **Date: 2082/04/27**

Title: Write a program to display the number of page fault for user input reference string and frame size using LRU.

Definition:

LRU stands for Least Recently Used. It is a page replacement algorithm that replaces the page that hasn't been used for the longest period of time. It is based on the assumption that page used recently will likely be used again soon.

How OPR Works:

For each page reference:

- If the page is already in memory → Page Hit
- If not → Page Fault

If memory is full:

- Identify the page in memory that was least recently used.
- Replace that page with the new one.

IDE : DEV C++

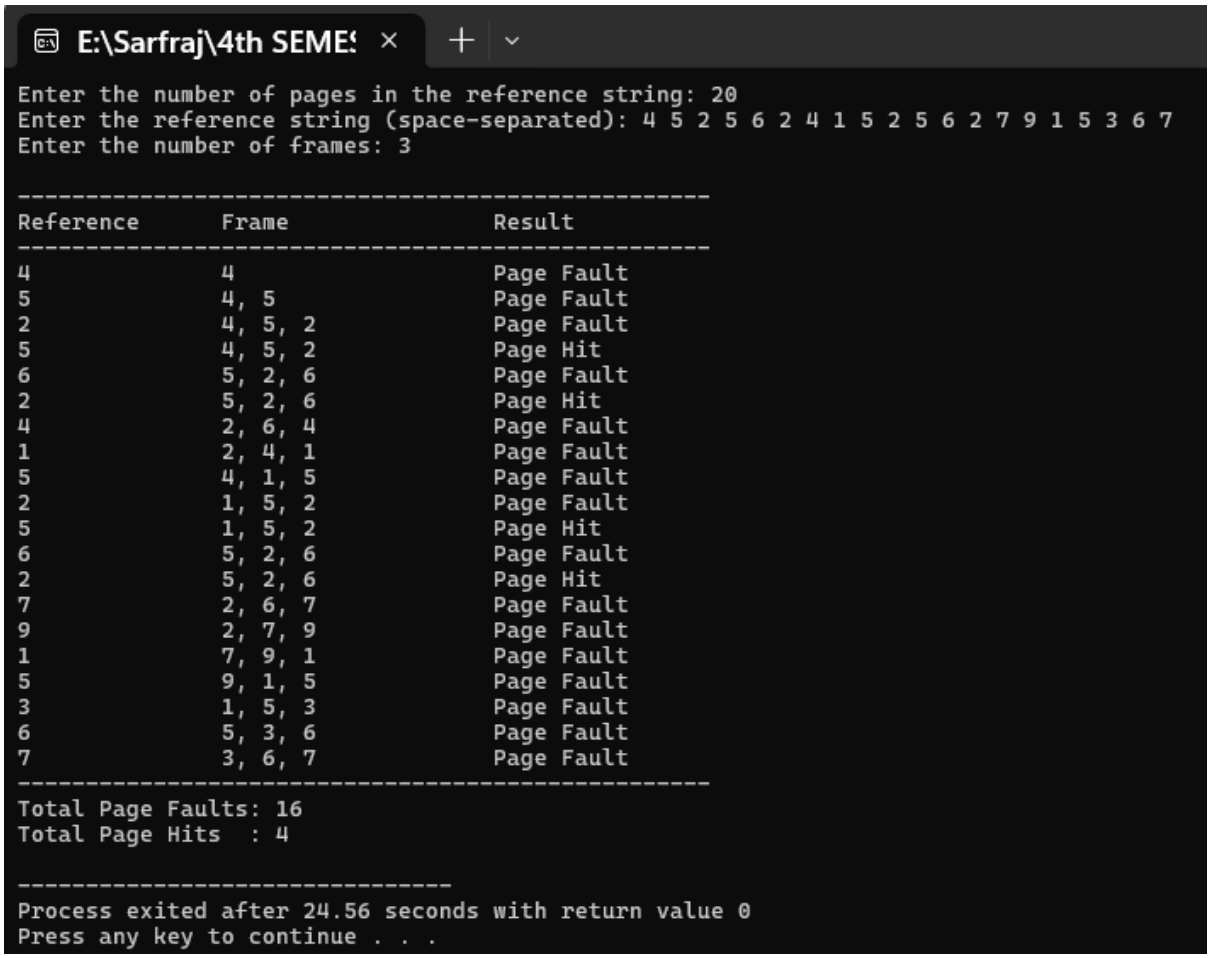
Source code:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <iomanip>
#include <sstream>

using namespace std;

// Function to convert int to string for older compilers
string intToString(int num) {
    stringstream ss;
    ss << num;
    return ss.str();
}

// Function to update recently used pages
void updateLRU(vector<int>& usageOrder, int page) {
    vector<int>::iterator it = find(usageOrder.begin(), usageOrder.end(), page);
    if (it != usageOrder.end()) {
        usageOrder.erase(it);
    }
    usageOrder.push_back(page); // move to the back (most recently used)
}

int main() {
    int n, frameSize;
    vector<int> referenceString;
    vector<int> frameVec;
    vector<int> usageOrder; // keeps track of LRU
    int pageFaults = 0;

    // Input
    cout << "Enter the number of pages in the reference string: ";
    cin >> n;

    cout << "Enter the reference string (space-separated): ";
    for (int i = 0; i < n; ++i) {
        int page;
        cin >> page;
        referenceString.push_back(page);
    }

    cout << "Enter the number of frames: ";
    cin >> frameSize;

    // Header
    cout << "\n-----------------------------------------------\n";
    cout << left << setw(15) << "Reference"
         << setw(20) << "Frame"
```

```
     << setw(15) << "Result" << endl;
cout << "-------------------------------------------------\n";

for (int i = 0; i < n; ++i) {
    int currentPage = referenceString[i];
    string result;

    // Check if page is in frame
    vector<int>::iterator it = find(frameVec.begin(), frameVec.end(), currentPage);

    if (it != frameVec.end()) {
        // Page Hit
        result = "Page Hit";
    } else {
        // Page Fault
        result = "Page Fault";
        pageFaults++;

        if ((int)frameVec.size() < frameSize) {
            frameVec.push_back(currentPage);
        } else {
            // Remove least recently used page
            int lruPage = usageOrder.front();
            usageOrder.erase(usageOrder.begin());

            vector<int>::iterator removeIt = find(frameVec.begin(), frameVec.end(), lruPage);
            if (removeIt != frameVec.end()) {
                frameVec.erase(removeIt);
            }

            frameVec.push_back(currentPage);
        }
    }

    // Update usage history
    updateLRU(usageOrder, currentPage);

    // Print output for this step
    cout << left << setw(15) << currentPage;

    // Frame contents
    string frameContent;
    for (size_t j = 0; j < frameVec.size(); ++j) {
        frameContent += intToString(frameVec[j]);
        if (j != frameVec.size() - 1) frameContent += ", ";
    }

    cout << setw(20) << frameContent
         << setw(15) << result << endl;
}
```

```
    // Final Summary
    cout << "-------------------------------------------------\n";
    cout << "Total Page Faults: " << pageFaults << endl;
    cout << "Total Page Hits  : " << n - pageFaults << endl;

    return 0;
}
```

Output:


E:\Sarfraj\4th SEME
Enter the number of pages in the reference string: 20
Enter the reference string (space-separated): 4 5 2 5 6 2 4 1 5 2 5 6 2 7 9 1 5 3 6 7
Enter the number of frames: 3
Reference Frame Result
4 4 Page Fault
5 4, 5 Page Fault
2 4, 5, 2 Page Fault
5 4, 5, 2 Page Hit
6 5, 2, 6 Page Fault
2 5, 2, 6 Page Hit
4 2, 6, 4 Page Fault
1 2, 4, 1 Page Fault
5 4, 1, 5 Page Fault
2 1, 5, 2 Page Fault
5 1, 5, 2 Page Hit
6 5, 2, 6 Page Fault
2 5, 2, 6 Page Hit
7 2, 6, 7 Page Fault
9 2, 7, 9 Page Fault
1 7, 9, 1 Page Fault
5 9, 1, 5 Page Fault
3 1, 5, 3 Page Fault
6 5, 3, 6 Page Fault
7 3, 6, 7 Page Fault
Total Page Faults: 16
Total Page Hits : 4
Process exited after 24.56 seconds with return value 0
Press any key to continue . . .