Experiment Number: 10

Problem Statement: **Write a program to calculate the number of page faults for a reference string for the following page replacement algorithms:**

**a) FIFO     b) LRU     c) Optimal**

NAME: Aadesh Chawla ROLLNO: 12

CLASS: TY-IT-A BATCH: B1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Code:

#include <iostream>

#include <vector>

#include <limits>

#include <sstream>

using namespace std;

// FIFO (First-In, First-Out) Page Replacement Algorithm

int fifo(vector<int>& reference, int frames) {

    vector<int> frame(frames, -1);

    int page\_faults = 0;

    int index = 0;

    for (int i = 0; i < reference.size(); i++) {

        int page = reference[i];

        int is\_fault = 1;

        for (int j = 0; j < frames; j++) {

            if (frame[j] == page) {

                is\_fault = 0;

                break;

            }

        }

        if (is\_fault) {

            page\_faults++;

            frame[index] = page;

            index = (index + 1) % frames;

        }

    }

    return page\_faults;

}

// LRU (Least Recently Used) Page Replacement Algorithm

int lru(vector<int>& reference, int frames) {

    vector<int> frame(frames, -1);

    vector<int> counter(frames, 0);

    int page\_faults = 0;

    for (int i = 0; i < reference.size(); i++) {

        int page = reference[i];

        int is\_fault = 1;

        for (int j = 0; j < frames; j++) {

            if (frame[j] == page) {

                is\_fault = 0;

                counter[j] = i;

                break;

            }

        }

        if (is\_fault) {

            page\_faults++;

            int min\_count = counter[0], replace = 0;

            for (int j = 1; j < frames; j++) {

                if (counter[j] < min\_count) {

                    min\_count = counter[j];

                    replace = j;

                }

            }

            frame[replace] = page;

            counter[replace] = i;

        }

    }

    return page\_faults;

}

// Optimal Page Replacement Algorithm

int optimal(vector<int>& reference, int frames) {

    vector<int> frame(frames, -1);

    int page\_faults = 0;

    for (int i = 0; i < reference.size(); i++) {

        int page = reference[i];

        int is\_fault = 1;

        for (int j = 0; j < frames; j++) {

            if (frame[j] == page) {

                is\_fault = 0;

                break;

            }

        }

        if (is\_fault) {

            page\_faults++;

            int replace = -1;

            for (int j = 0; j < frames; j++) {

                int k;

                for (k = i + 1; k < reference.size(); k++) {

                    if (reference[k] == frame[j]) {

                        if (k > replace)

                            replace = k;

                        break;

                    }

                }

                if (k == reference.size()) {

                    replace = j;

                    break;

                }

            }

            frame[replace] = page;

        }

    }

    return page\_faults;

}

int main() {

    int frames, pages;

    cout << "Enter the number of frames: ";

    cin >> frames;

    cout << "Enter the number of pages: ";

    cin >> pages;

    cin.ignore(); // Consume the newline character.

    vector<int> reference;

    cout << "Enter the reference string (space-separated): ";

    string input;

    getline(cin, input);

    istringstream iss(input);

    int page;

    while (iss >> page) {

        reference.push\_back(page);

    }

    int fifo\_faults = fifo(reference, frames);

    int lru\_faults = lru(reference, frames);

    int optimal\_faults = optimal(reference, frames);

    cout << "Total page faults using FIFO: " << fifo\_faults << endl;

    cout << "Total page faults using LRU: " << lru\_faults << endl;

    cout << "Total page faults using Optimal: " << optimal\_faults << endl;

    return 0;

}

Output:

