…for a complete human being

Texas

International College

Lab No: 3 **Date: 2082/**

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

LRU replaces the page that has not been used for the longest time in the past. It approximates the Optimal algorithm but works in real systems because it only relies on past usage. It Uses the concept of recency to decide which page to replace. Can be implemented using counters, stacks, or linked lists to track usage.

Algorithm:

Step 1: Initialize all frames as empty.

Step 2: Read the reference string.

Step 3: For each page request:

- If the page is already in frame → Page Hit and update its recent usage.
- If the page is not in frame → Page Fault:
    - If free space → insert the page.
    - Else → find the page that was least recently used (used longest ago) and replace it.

Step 4: Update page hit/fault counters after each request.

Step 5: Repeat Step 3 for all pages and display the results.

Language: C++

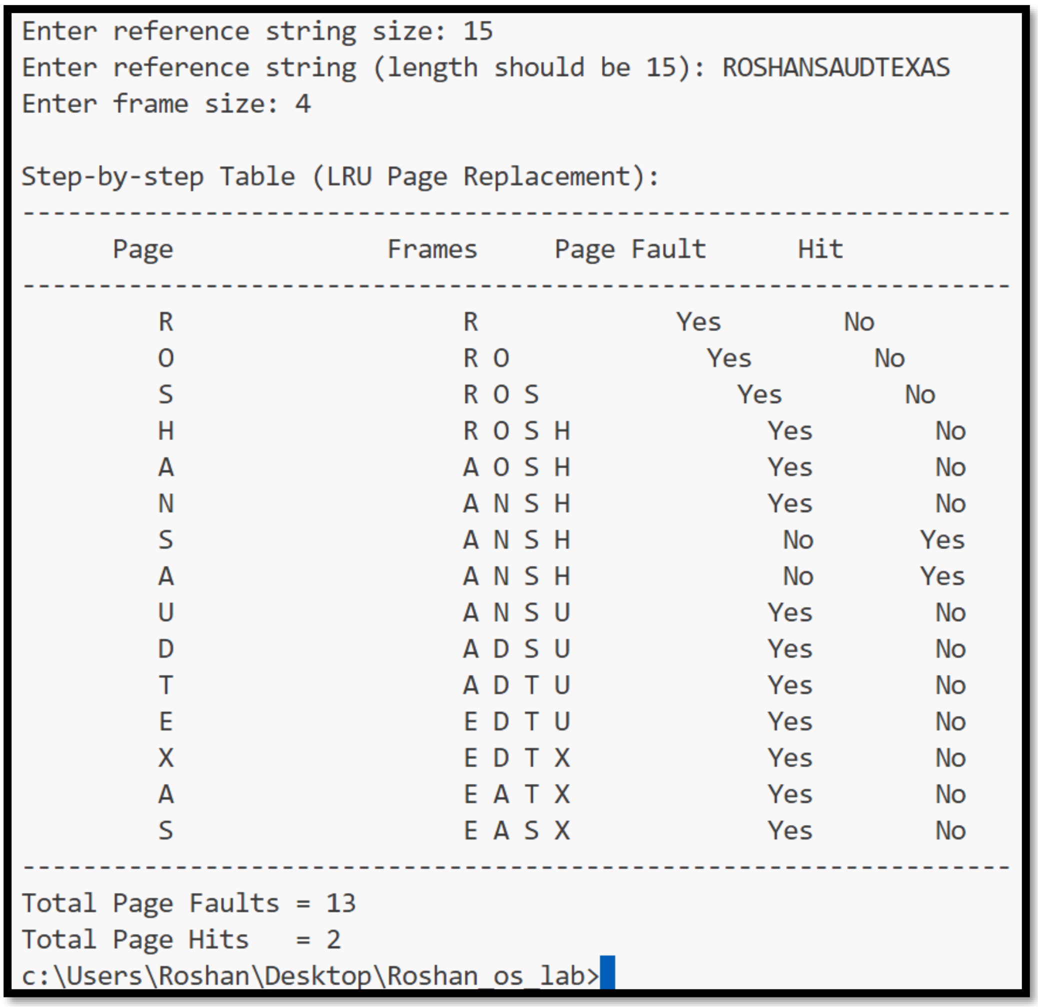
IDE: VS Code

Code:

```
#include <iostream>

#include <vector>

#include <iomanip>

using namespace std;


void lruPageReplacement(const string &referenceString, int frameSize) {

    vector<char> frames;          // store characters in frames

    vector<int> lastUsed;         // parallel vector to store last used index

    int pageFaults = 0, pageHits = 0;

    cout << "\nStep-by-step Table (LRU Page Replacement):\n";

    cout << "------------------------------------------------------------\n";

    cout << setw(10) << "Page"

         << setw(20) << "Frames"

         << setw(15) << "Page Fault"

         << setw(10) << "Hit\n";

    cout << "------------------------------------------------------------\n";

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

        char page = referenceString[i];

        bool found = false;

        // Check if page already in frames

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

            if (frames[j] == page) {

                found = true;

                lastUsed[j] = i; // update last used index
                break;
            }
        }
        if (!found) { // Page fault
            if ((int)frames.size() < frameSize) {
                frames.push_back(page);
                lastUsed.push_back(i);
            } else {
                // Find least recently used page
                int lruIndex = 0;
                for (int j = 1; j < frameSize; j++) {
                    if (lastUsed[j] < lastUsed[lruIndex])
                        lruIndex = j;
                }
                frames[lruIndex] = page;
                lastUsed[lruIndex] = i;
            }
            pageFaults++;
        } else {
            pageHits++;
        }
        // Print current step
        cout << setw(10) << page << setw(20);
        for (char f : frames) cout << f << " ";
        cout << setw(15) << (found ? "No" : "Yes")
```

```
            << setw(10) << (found ? "Yes" : "No") << "\n";
    }
    cout << "----------------------------------------------------------------------------\n";
    cout << "Total Page Faults = " << pageFaults << endl;
    cout << "Total Page Hits   = " << pageHits;
}
int main() {
    int frameSize, refSize;
    string referenceString;
    cout << "Enter reference string size: ";
    cin >> refSize;
    cout << "Enter reference string (length should be " << refSize << "): ";
    cin >> referenceString;
    cout << "Enter frame size: ";
    cin >> frameSize;
    if ((int)referenceString.size() != refSize) {
        cout << "Error: reference string size does not match input length!\n";
        return 1;
    }
    lruPageReplacement(referenceString, frameSize);
    return 0;
}
```

Output:



```
Enter reference string size: 15
Enter reference string (length should be 15): ROSHANSAUDTEXAS
Enter frame size: 4

Step-by-step Table (LRU Page Replacement):
-----------------------------------------------------------------
      Page                 Frames      Page Fault      Hit
-----------------------------------------------------------------
         R                   R             Yes         No
         O                   R O             Yes         No
         S                   R O S             Yes         No
         H                   R O S H             Yes         No
         A                   A O S H             Yes         No
         N                   A N S H             Yes         No
         S                   A N S H              No         Yes
         A                   A N S H              No         Yes
         U                   A N S U             Yes         No
         D                   A D S U             Yes         No
         T                   A D T U             Yes         No
         E                   E D T U             Yes         No
         X                   E D T X             Yes         No
         A                   E A T X             Yes         No
         S                   E A S X             Yes         No
-----------------------------------------------------------------
Total Page Faults = 13
Total Page Hits   = 2
c:\Users\Roshan\Desktop\Roshan_os_lab>
```

Conclusion:

LRU effectively reduces page faults compared to FIFO by considering recent usage. While it may not always be as optimal as the theoretical Optimal algorithm, it is practical and widely used in real systems.