

**Lab No: 2 Date: 2082/**

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

Optimal Page Replacement replaces the page that will not be used for the longest time in the future. It yields the minimum possible page faults for a given reference string, so it’s mainly used as a theoretical benchmark (not practical in real systems because the future isn’t known).

Algorithm:

Step 1: Initialize frames as empty.

Step 2: Read the reference string.

Step 3: For each page request:

* If page is in frame → Page Hit.
* If not in frame → Page Fault:
  + If free space → insert page.
  + Else → find the page in frame that will be used farthest in the future (or not used again) and replace it.

Step 4: Update page hit/fault count.

Step 5: Repeat until all pages are processed and display results.

**Language**: C++

**IDE**: VS Code

**Code:**

**#include <iostream>**

**#include <vector>**

**#include <iomanip>**

**using namespace std;**

**void optimalPageReplacement(const string &referenceString, int frameSize) {**

**vector<char> frames;**

**int pageFaults = 0, pageHits = 0;**

**int refLen = referenceString.length();**

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

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

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

**<< setw(20) << "Frames"**

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

**<< setw(15) << "Page Hit\n";**

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

**for (int i = 0; i < refLen; ++i) {**

**char currentPage = referenceString[i];**

**bool found = false;**

**// Check if current page is already in frame**

**for (char f : frames) {**

**if (f == currentPage) {**

**found = true;**

**break;**

**}**

**}**

**if (!found) { // Page fault**

**if ((int)frames.size() < frameSize) {**

**frames.push\_back(currentPage);**

**} else {**

**// Find the page in frames that will not be used for the longest time**

**int indexToReplace = -1;**

**int farthest = -1;**

**for (int j = 0; j < frameSize; ++j) {**

**int k;**

**for (k = i + 1; k < refLen; ++k) {**

**if (referenceString[k] == frames[j])**

**break;**

**}**

**if (k > farthest) {**

**farthest = k;**

**indexToReplace = j;**

**}**

**}**

**// Replace the chosen page**

**frames[indexToReplace] = currentPage;**

**}**

**pageFaults++;**

**} else {**

**pageHits++;**

**}**

**// Print current step**

**cout << setw(10) << currentPage << setw(20);**

**for (char f : frames) cout << f << " ";**

**cout << setw(15) << (found ? "No" : "Yes")**

**<< setw(15) << (found ? "Yes" : "No") << "\n";**

**}**

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

**cout << "Total Page Faults = " << pageFaults << endl;**

**cout << "Total Page Hits   = " << pageHits;**

**}**

**int main() {**

**int refSize;**

**string referenceString;**

**int frameSize;**

**cout << "Enter reference string size: ";**

**cin >> refSize;**

**cout << "Enter reference string: ";**

**for (int i = 0; i < refSize; ++i) {**

**char page;**

**cin >> page;**

**referenceString += page;**

**}**

**cout << "Enter frame size: ";**

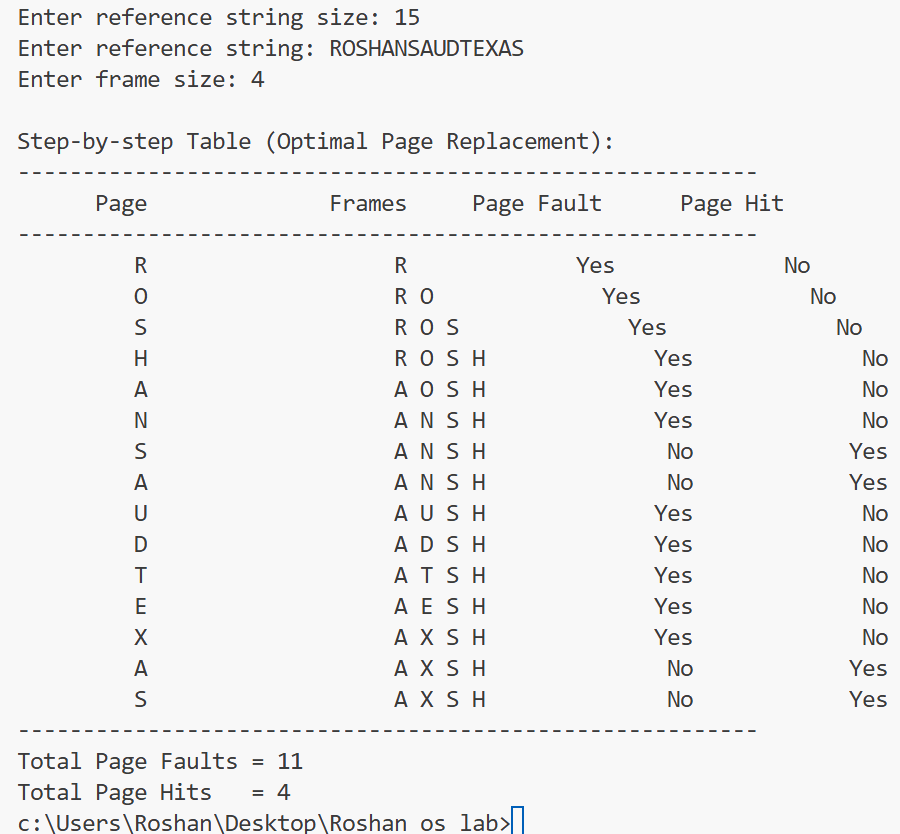
**cin >> frameSize;**

**optimalPageReplacement(referenceString, frameSize);**

**return 0;**

**}**

**Output:**

****

**Conclusion:**

The OPR gives the lowest possible page faults and never exhibits Belady’s anomaly, making it the gold-standard benchmark for evaluating other algorithms. However, it’s not implementable in practice because it requires future knowledge of references; hence systems use approximations like LRU instead.