

/* 5. Compare the performance in terms of number of page faults for the algorithm that suffers from Belady's anomaly with the one that replaces a page that has not been used for the longest period of time. */

// Purpose of a computer system is to execute programs - programs, together with the data they access, must be at least partially in main memory during execution

// General-purpose computer must keep several processes in memory
// Many memory-management algorithms exist, each having its own advantages and disadvantages, tend to require an entire process be in memory before it can execute

// Virtual memory is a technique that allows the execution of processes that are not completely in memory
// Major advantage of this scheme is that programs can be larger than physical memory

// Process address space is broken into blocks of the same size called pages, Physical memory into fixed-sized blocks called frames
// Size of the process is measured in the number of pages

// Strategy to load pages into frames only as they are needed - is demand paging
// Demand paging is commonly used in virtual memory systems

// Lazy pager swaps a page into memory (from secondary) only when that page is needed

// Whenever a new page is referred and not present in memory, page fault occurs
// and Operating System replaces one of the existing pages with newly needed page

// Different page replacement algorithms suggest different ways to decide which page to replace - goal is to reduce number of page faults
// First-in First-out(FIFO), Optimal, Least Recently Used(LRU)

// FIFO page replacement algorithm associates with each page the time when that page was brought into memory. When a page must be replaced, the oldest page is chosen

// Optimal page replacement replaces the page that will not be used for the longest period of time, but difficult to implement as it requires future knowledge of the reference string, similar problem as in SJF CPU-scheduling algorithm

// LRU - uses the recent past as an approximation of the near future, replaces the page that has not been used for the longest period of time

// We would expect that giving more memory(frames) to a process would improve its performance - that is page fault and number of frames are inversely proportional

// But for some page-replacement algorithms, the page-fault rate may increase as the number of allocated frames increases - called as Belady's anomaly

// FIFO suffers from Belady's anomaly
// Optimal and LRU Page Replacement never suffers from Belady's anomaly

// A page that has not been used for the longest period of time is Least Recently Used

// Plot the graph of Number of frames versus Page faults

```

#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int referenceString[]={1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5};
int lengthOfReferenceString = sizeof(referenceString) / sizeof(int);
int pagePresentInFrames; // Flag to indicate presence of particular page in frame
int pageFaults;          // To count page faults

void printPagesInFrames( int frame[], int numberOfFrames )
{ // Function prints current pages in frames, -1 if no page in frame
    int k;
    for( k=0; k < numberOfFrames; k++) printf("\t %d",frame[k] );
}

int findIndexOfLeastRecentlyUsed( int lruCounter[], int numberOfFrames )
{ // Function return index of Least Recently Used page
    int lruValue = -1;
    int indexOfLRU = 0;
    int i;
    for( i=0; i<numberOfFrames; i++)
        if( lruCounter[i] == -1 ) // Return index of first occurrence of free frame
            return i;
        else if( lruCounter[i] > lruValue ) // else find frame with highest counter
        { // hence least recently used
            lruValue = lruCounter[i];
            indexOfLRU = i;
        }

    return indexOfLRU;
}

//void pageReplacement( numberOfFrames , 0 for FIFO and 1 for LRU )
void pageReplacement( int numberOfFrames , int pageReplacementAlgorithm)
{ // counter For : FIFO : highest value will be First In; And for
    int lruCounter[5] = {-1, -1, -1, -1, -1}; // LRU : highest value will be least recently used
    int frame[5] = {-1, -1, -1, -1, -1}; // To remember page in frame, initialized as unallocated
    int i, j, k;

    pageFaults = 0; // Initialize page faults = 0

    printf(" using %d frames, initial frames = ", numberOfFrames);
    printPagesInFrames( frame, numberOfFrames );

    printf("\n Page in reference string\t\t Pages Frames");
    for( i=0; i<lengthOfReferenceString; i++) // For each page in reference string
    {
        printf("\n\t\t %d\t\t",referenceString[i]); //print page that will be allocated frame

        pagePresentInFrames=0; // Assume page is not present in frames
        for( k=0; k < numberOfFrames; k++) // For each frame
            if( frame[k] == referenceString[i] ) // Check if page is present in frames
            {
                pagePresentInFrames=1; // Page exists in frames

                if( pageReplacementAlgorithm == 0 ) // 0 for FIFO and 1 for LRU

```

```

        { // Its FIFO, do not re initialize counter to zero, but increment count
          lruCounter[k]++; // Though recently used, here highest count implies First In
        }
        else
        { // Its LRU, re initialize counter
          lruCounter[k] = 0; // page used, hence reinitialize counter as recently used
        }
      }
      else if( lruCounter[k] != -1 )
        lruCounter[k]++; // It is different page, update recently used counter

    if ( pagePresentInFrames == 0 ) // If page was not present in frames
    {
      j = findIndexOfLeastRecentlyUsed( lruCounter, numberOfFrames );
      frame[j]=referenceString[i]; // allocate j th frame to page referenceString[i]
      lruCounter[j] = 0; // initialize j th counter as recently used
      pageFaults++; // Increment page faults
      printPagesInFrames( frame, numberOfFrames );
    }
  }
  printf("\n Page Faults are = %d\n",pageFaults);
}

int main() // read the number of pages, frames and reference string from user
{
  int i;
  printf("\n Reference string = "); // print reference string
  for( i=0; i < lengthOfReferenceString; i++)    printf(" %d",referenceString[i]);

  printf("\n FIFO Page Replacement"); // Try with 3, 4 and 5 frames
  pageReplacement( 2 , 0 ); // pageReplacement( numberOfFrames, 0 for FIFO )

  printf("\n LRU Page Replacement"); // Try with 3, 4 and 5 frames
  pageReplacement( 2 , 1 ); // pageReplacement( numberOfFrames, 1 for LRU )

  return 0; // Try 3, 2, 1, 0, 3, 2, 4, 3, 2, 1, 0, 4, 3
}/* also 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0 , 1, 7, 0, 1 */

```