**CSCI 331 Project Four**

**Note:** This is a project from your Zybooks textbook.

**Project objective**

Compare the performance of page replacement algorithms for fixed numbers of frames: optimal, FIFO, LRU, second chance.

**Description**

The simulation generates reference strings based on typical characteristics of program execution. The strings are then used to compare the performance of different algorithms in terms of the page faults generated.

* The virtual memory of a process consists of P pages, numbered 0 through P −1.
* A reference string RS is a sequence of integers in the range [0 : P − 1]. Each element, p, of RS represents one reference to page p.
* The physical memory of the system is approximated as an integer array M[F], where F is the number of frames. Each element M[f] represents one frame and contains the number p of the page currently residing in the frame f.

The main distinguishing property of a reference string RS is the degree of locality. A reference string constructed as a sequence of random numbers distributed uniformly over the range [0 : P - 1] has no locality. Such a string represents a situation where too many processes are running concurrently and the lack of locality results in thrashing.

Most programs display a high degree of locality, which can be modeled as follows:

* Consecutive pages are drawn from a small region within [0 : P - 1], referred to as the current locus of reference, L.
* L is defined by a starting position, s, within [0 : P - 1] and size, e.  
  In reality, L is not a contiguous region since pages from at least three regions (code, data, and stack) are being accessed. For studying page-replacement algorithms, however, only the size of L is of interest and thus a contiguous locus of reference is adequate.
* The locus L gradually shifts to the right, which emulates sequential execution of the program. The rate of the motion is governed by a constant, m, which determines how many pages are chosen from L before L is shifted to the right.
* At random time points, L moves to a new location, s, within [0 : P - 1], chosen at random. The change represents the occasional transition of the program to a new area, resulting from a function call or a branch instruction.

Using the above assumptions, a reference string is generated as follows:

select parameters for the new RS: /\* RS is an initially empty file \*/

virtual memory size: P /\* A large integer. Ex: with address size

of 32 bits and page size of 4096, P = 2^20 \*/

starting location: s /\* Initial s may be 0 \*/

size of L: e /\* Small integer representing current locus \*/

rate of motion: m /\* Each page in L is typically referenced only a few

hundred times. Ex: m = 200 \*/

probability of transition: t /\* A real number close to 0 since transitions are rare \*/

repeat until RS of desired length is generated

select m random numbers in the range [s:s+e] /\* Process is executing within current locus \*/

append the numbers to RS /\* RS is a file that grows with each iteration \*/

generate random number r in the range [0:1]

if (r < t), generate new s /\* Process transitions to a new location s \*/

else increment s by 1 (modulo P) /\* Process remains within current locus \*/

**Assignment**

* Implement the program to generate reference strings.
* Generate sets of reference strings by varying the input parameters e and t, to mimic different program behaviors (high vs low locality, fast vs slow changing locus).
* Implement two or more of the page replacement algorithms.
* Record and plot the numbers of page faults generated by different algorithms against t or e.