CSCD 340
Lab 10

Write a C program that translates logical address to physical addresses. The program will read from a file containing logical addresses and will translate each logical address to its corresponding physical address. The program will also keep track of page hits and page misses.

There are 2 files required for this program.

- File 1 – setup.txt – a text file that contains in this order
    - An int that represents the total size of the virtual address space (Example: 1048576 would represent a virtual address space that is 1 MB in size or 20 bits)
    - An int that represent the size of a single page/frame (Example: 256 would be a page is 256 bytes in size or 8 bits)
    - An int that represents the total size of the physical address space (Example: 65536 would represent a physical memory space that is 64 KB in size or 16 bits)
    - You will have to do the math. In this case we have you have 4096 pages, each page is 256 bytes in size, and you have 256 frames each frame is 256 in size)
- File 2 – the virtual addresses to be translated. This file contains several base ten, 32-bit integers that represent logical addresses.
- You may hardcode setup.txt, you must prompt for the name of File 2.

NOTE: you are guaranteed valid powers of 2 numbers and **the Happy Part of Stuland**

<u>To Do</u>

1. Create one array/linked list that contains the number of pages in the logical address space.
    - This array will be an array of structures. The structure will be named Page Table Entry (PTE).
    - The PTE will contain the page frame it is mapped to as a decimal number, a referenced int that will be referenced anytime the page is referenced, a modified bit for future use, and any other items you deem necessary.
2. Read the logical address from the file, determine the page for that address
    - If the page is mapped, the frame number is obtained from the page table entry within the page.
    - If the page is not mapped (a miss) then a page fault happens and the page table entry must be loaded with a page frame number that is not currently being used.
    - If there is no available page frame number, then evict page 0, then page 1, then page 2, etc.
    - For example say page 3 is referenced as the first page
        - This reference will cause a miss
        - Page 3 will then be mapped to page frame 0
        - The reference bit will be set
        - The physical address will be displayed
    - Later on page 3 is again reference
        - This reference will be a hit
        - You will increment the referenced count
        - The physical address will be displayed

Test File

- I provide the file `test.txt`, which contains ints representing logical addresses ranging from 0 through the size of the virtual address space.

Statistics

- Your program is to report the following per address
  - Virtual Address: the virtual address value
  - Page Number: the page number of the virtual address
  - If the reference was a hit or a miss
  - Page Frame Number: the page frame number
  - Physical Address: the physical address

- After all addresses have been processed, report the following
  - Number of translated addresses
  - Number of Page Faults
  - The Page Rate – number of page faults divided by the number of address translated
  - The Number of pages evicted
  - The page eviction rate

Future Work:

You will want to write your code as generic as possible. We will be revisiting this project to better understand the replacement algorithms.

TO TURN IN

A single zip file containing:

- All C files
  - ensure your program compiles and runs
  - The one that contains main will be called cscd340lab10.c
- All input files
  - setup.txt
  - test.txt/addresses.txt or whatever you called it
- A makefile with the target of lab10 (NOTE no makefile that compiles your code into the executable lab10 then NO POINTS)

Name your zip file your last name first letter of your first name lab10 (Example steinerslab10.zip)