**Lab 4 – FIFO Page Replacement Policies Observations**

For this lab, we are to write a program that simulates the FIFO page replacement algorithms, and the program is supposed to accept at least one numeric command-line parameter, which it will then be use as the number of available page frames.

The implementation of the FIFO simulator is done by using a queue with a linked list. We were provided with some .c and .h files and using the queue\_test.c file we test our functions. A skeleton of what our fifo.c should look like was given to us and we basically fill in the area where it says, ‘page replacement implementation go here’ for our fifo.c program. I changed a bit of the skeleton and replaced what needs to be in that area.

For the algorithm logic, it simultaneously reads each page entry and increment the line counter for each line. It also takes the line input of each line and search through the queue data structure. If the line is not found it increments the page faulter counter, and if the size of the queue data structure is less than the size of the user input; it adds the page number to the data structure. And if the queue data structure is greater than or equal to 100 of the user input that it removes the first node at the head of the data structure and insert the page number at the tail of the data structure.

**Chart of the test memory sizes between 100 and 1500 pages:  
Hit Rate vs. Cache Size**

From the chart at memory size/cache size of 100 – 900, we see that the hit rates increases/improves along with a larger cache size. However, as it reaches 1000 and above the hit rates plateaus and stayed at around 90.01%. I tested below 100 and they seem to indicate the same thing happening in between the cache size of 100-900. And if I go beyond the 1500 it emulates the same outcome as anywhere between the 1000 -1500 ranges. So, the cache sizes will hit a max size and will not read any more faults or hit rates higher than above for this file. I was told that the reason it plateaus is due to the distinct pages in accesses.txt that go up from a range of 0-998. Which means there are only 999 distinct pages. So, any cache size that is 1000 or more would have completely fulfilled the cache and the resulting access will always be a hit dur to the cache being filled. Which results in a consistent hit rate and number of faults.