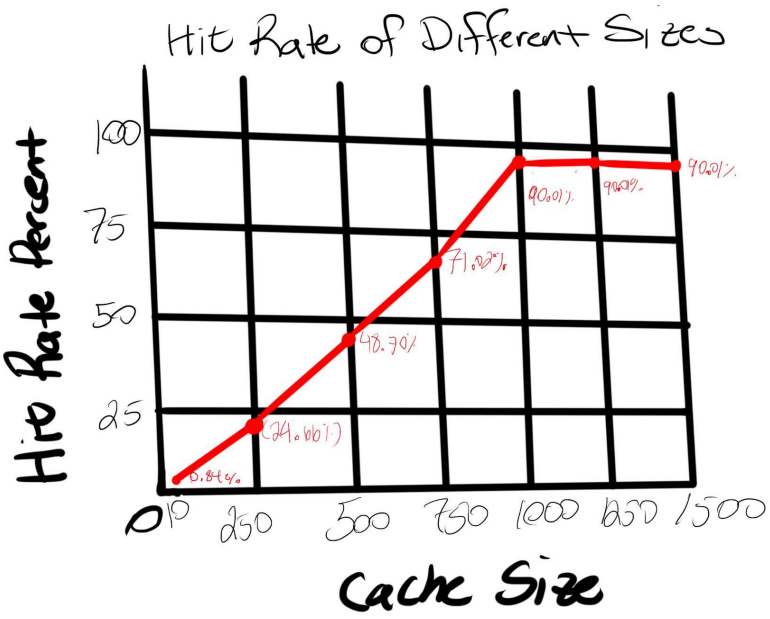
Yukio Rivera
3/29/22
Title: Lab 4
Description: Write up for Lab 4



I used the queue test file to figure out how to solve the homework problem of implementing FIFO type replacement to the skeleton.c file. In the image provided I show different hit rates that the program calculated. What I noticed was that the higher the cache size the less chance the algorithm would fault. This makes sense to me because if the cache size is larger there is a greater chance that the new page number is already queued in memory. I used the cache sizes 10 to 1500. 10 was below 1% and it grew at a rapid rate until about 1000. After 1000 it plateaued at 90.01%. I tried 1000, 1250, and 1500 showing the same 90.01% result. Its not shown in the diagram but I continued with greater numbers with the same result.

For the code I created the requested variables, totalFaults, totalAccesses, totalHits, and hitRate to provide the requested calculations. In the while loop I incremented totalAccesses to keep a counter of all the pages. Used the queue.c functions to recreate the FIFO algorithm using enqueue, to add to the cache, queue_find to check if the page is in the que, queue_length to check when the queue is full (size determined by the user) and deque to remove a page to make room for a new one. I created the queue using queue_create and cleared it from memory at the end of the program using queue_destroy. The hitRate calculation was done by dividing the totalHits by totalAccesses and multiplying the result by 100 to get the percent.
(`((float)totalHits/(float)totalAccesses) * 100)`)

Test Results:

100: 9.82%
200: 19.59%
300: 29.49%
400: 38.97%
500: 48.70%
600: 57.92%
700: 66.98%
800: 75.03%
900: 83.28%
1000: 90.01%
1100: 90.01%
1200: 90.01%
1300: 90.01%
1400: 90.01%
1500: 90.01%