Daniel Murga

1. When creating the hash table, a for loop is used based on the load factor, starting from 0.2 to 0.9, incrementing by 0.1. Within the loop, a nested for loop from 1 to 5 sets the seed using srand(i) and gets random numbers and places them into an array. It then creates a hashtable of size 600011 * load factor, starts the timer, populates the table, and calculates the duration of insertion for each table. It does this for all 3 hashtables and outputs the results to the terminal and calculates the average with the stored durations for each table. Each iteration increases the load factor lambda by 0.1 and does this process again for different sized tables based on lambda.
2. To generate the data for the tables, the for loop from 1 to 5 sets the seed in srand(i) based on the value of i. The next loop iterates from 0 to size (600011 * load factor) and populates an array called random with values using rand(). This array is used to populate the 3 hashtables.
3. Until reaching load factor 0.8, the closed hashtables were about 15 percent faster than the open hashtable, double hashing being slightly faster than quadratic probing. At lambda = 0.8 the open hashtable outperformed both of the closed hashtables by about 20%. Open was faster the higher the load factor and slower the smaller the value.
4. Once closed hashtables reach a certain load, the open hashtable begins to outperform both types of closed hashtables, with a significant difference in its speed compared to its slowness at lower load levels. Quadratic probing continually stayed in between its double counterpart and open, but was still slower at the 0.9 load factor. When using a hashtable with a small load factor, double hashing is the fastest implementation, but after lambda = 0.7, open hashing is the more appropriate choice to use when the table will be filled almost completely.