**Coding Assignment 6 Results**

CSE 3318

**Test 1**

**A.** How many rows are in your file/how many cells are in your hash table array?

*(put your answer here) 30*

**B.** How many of those cells contained the head of a linked list?

*(put your answer here) 17*

**C.** What percentage of the array is being used?

*(put your answer here) 56.5*

**D.** What is the length of the longest linked list?

*(put your answer here) 3*

**Test 2**

**A.** Did increasing the size of the hash table array give you different results than Test Question 1?

*(put your answer here) yes*

**B.** Explain why or why not.

*(put your answer here) the percentage of the array used decreased significantly (21 out of 60 is 35%)*

**Test 3**

**A.** How many rows are in your file/how many cells are in your hash table array?

*(put your answer here) 15*

**B.** How many of those cells contained the head of a linked list?

*(put your answer here) 12*

**C.** What percentage of the array is being used?

*(put your answer here) 80*

**D.** How did decreasing the size of the hash table array affect the percentage of the array that filled?

*(put your answer here) the percentage of the array filled increased significantly.*

**E.** Did your hash table get any linked lists that were longer than in Test 1? Why or why not?

*(put your answer here) Yes, it did (longest is 5). It is because all these same number of elements are fitted into a much smaller place, so they keep getting added to the linked list by creating new nodes.*

**Test 4**

**A.** What was your average search time when your HASHTABLESIZE matched the number of records in the file?

*(put your answer here) 10.*

**Test 5**

**A.** What was your average search time when your HASHTABLESIZE was set to 1?

*(put your answer here) 8.6*

**Test 6**

**A.** What was your average search time when your HASHTABLESIZE was set to 1 and you only searched for the last record of your input file?

*(put your answer here) 9.8*

**B.** Was this average different from your answer to Test 5. If yes, why?

*(put your answer here) Yes, slightly different. And this could be because I’m searching for the same element every time so it will have to go to that specific position everytime, but when doing different searches, it goes to diferent places each time, so it might take lesser time.*

**Bonus Question**

If your program was using Open Addressing rather than Separate Chaining, then how many cells of the hash table array would be used when HASHTABLESIZE is set to the number of lines in the file? How did you calculate this number? Show/explain your answer.  
  
If my program was using Open Addressing rather than Separate Chaining, then all cells of the hash table would be filled when HASHTABLESIZE is set to the number of lines in the file. Example, if my HASHTABLESIZE = 30 = number of lines in file, then all 30 cells of the array would be filled. This is because in Open Addressing, when there is a collision at a cell, then the program looks for the next empty cell and puts this new value there. And if the end of the array is reached while doing so, the program will go to the beginning of the array and find the first empty cell ad put the new value there. As such, all cells of the array will be filled.