Data Structures (810:052) Lab 10 - Hashing Name:_____

Objective: To experiment with searching and get a feel for the performance of hashing.

To start the lab: Download and unzip the file lab11.zip

Part A:

a) Open and run the timeBinarySearch.py program that times the binarySearch algorithm imported from binarySearchIterativeLocation.py. Obverse that it creates a list, evenList, that holds 10,000 sorted, even values (e.g., evenList = [0, 2, 4, 6, 8, ..., 19996, 19998]). It then times the searching for target values from 0, 1, 2, 3, 4, ..., 19998, 19999 so half of the searches are successful and half are unsuccessful. How long does it take to binary search for target values from 0, 1, 2, 3, 4, ..., 19998, 19999?

b) Open and run the timeListDictSearch.py program that times the ListDict dictionary ADT in dictionary.py. The ListDict implementation uses a single Python list for storing dictionary entries. The timeListDictSearch.py program adds the 10,000 even values (i.e., 0, 2, 4, 6, 8, ..., 19996, 19998) to a ListDict object, and then times the searching for target values from 0, 1, 2, 3, 4, ..., 19998, 19999 so half of the searches are successful and half are unsuccessful. How long does it take to search for target values from 0, 1, 2, 3, 4, ..., 19998, 19999 in the ListDict?

c) Open and run the timeHashDictSearch.py program that times the HashDict dictionary ADT in dictionary.py. The HashDict implementation uses chaining with linked-lists for buckets. The timeHashDictSearch.py program adds the 10,000 even values (i.e., 0, 2, 4, 6, 8, ..., 19996, 19998) to a HashDict with 10,000 buckets (i.e., load factor of 1.0), and then times the searching for target values from 0, 1, 2, 3, 4, ..., 19998, 19999 so half of the searches are successful and half are unsuccessful. How long does it take to search for target values from 0, 1, 2, 3, 4, ..., 19998, 19999 in the HashDict?

d) Explain the relative performance results of searching using binary search, a ListDict, and a HashDict.

e) Experiment with changing the load factor of the HashDict between 0.2 and 0.9 by editting and rerun the timeHashDictSearch.py program. Completing the following table:

	Load Factor						
	0.2	0.4	0.6	0.8	1.0	2.0	10.0
Execution time with 10,000 even items in hash table (seconds)							
Hash table size							

f) Why does the performance of the HashDict degrade slowly as the load factor increases?

g) Implement and test the `HashDict` methods `keys()` and `values()`. `keys()` returns a Python list containing all of the keys stored in the `HashDict`. Similarly, `values()` returns a list of all the values.

After you have completed the above timings, questions, and coding, raise your hand and explain your answers.

Part B:

a) Open and run the timeHashSearch.py program that times the HashTable in hashtable.py. The HashTable implementation uses open-address hashing. The timeHashSearch.py program adds the 10,000 even values (i.e., 0, 2, 4, 6, 8, ..., 19996, 19998) to a HashTable of size 50,000 (i.e., load factor of 0.2) using linear probing for rehashing. It times the searching for target values from 0, 1, 2, 3, 4, ..., 19998, 19999 so half of the searches are successful and half are unsuccessful. How long does it take to search for target values from 0, 1, 2, 3, 4, ..., 19998, 19999 in the HashTable with load factor of 0.2?

b) Experiment with changing the load factor of the HashTable between 0.2 and 0.9 by editting and rerun the timeHashSearch.py program. Completing the following table:

Linear Probing	Load Factor							
	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Execution time with 10,000 items in hash table (seconds)								

c) Explain why the performance of the hash table with linear probing degrades so badly at high load factors.

d) In timeHashTable.py modify the construction of evenHashTable so it uses quadratic probing instead of linear probing (i.e., evenHashTable = HashTable(int(testSize/loadFactor), lambda x : x**, False**))**.** Experiment with changing the load factor of the HashTable using quadratic probing between 0.2 and 0.9 by editting and rerun the timeHashSearch.py program. Completing the following table:

Quadratic Probing	Load Factor							
	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Execution time with 10,000 items in hash table using quadratic probing (seconds)								

e) Explain why quadratic probing performs better than linear probing.

After you have performed the timings and answered the questions, raise your hand and explain your answers.