COMP 2160 Programming Practices
Assignment 3

Due Date Friday, November 18th at 11:59pm

Notes

- Please follow the programming standards; not doing so will result in a loss of marks.
- Your assignment **code** must be handed in electronically.

Question 1

In this problem, you will create an implementation of the Table abstract data type. A Table is a data type that allows for a few simple operations on a collection of data:

1. Insertion
2. Removal
3. Searching for presence
4. Iteration

Using the provided table interface `table.h` and the sample linked list code `linkedList.c`, complete an implementation of the Table ADT. Make sure that you apply the concepts of design by contract (DbC) to your implementation.

Once you have fully implemented the table, create a `main.c` file that implements a testing framework for your table.

Your table implementation must ensure that **values inserted are unique, and internally sorted** within a linked list. This is an *implementation detail* of your table, and not part of the public interface. Be sure to **only** test what is known from the public interface.

Make sure you test with assertions turned off. Boundary conditions should not cause the program to crash. Remember that to turn off assertions you can compile with `-DNDEBUG`.

Include a `Makefile` (along with all of your source files) for your implementation.

Question 2

You have been provided with a Set ADT developed by a third party. This consists of a header file, `set.h`, defining the interface and the object file, coming soon (as a zip file), compiled for use in our Mac lab. You must implement a complete unit test suite to validate all of the functionality defined in the header file.

Requirements/Notes:

1. Hand in your `main.c` file (containing your test suite implementation) and a `Makefile` to compile your code and link it with `set.o`.
2. You will be provided with, at least, two different object file implementations of Set. Include a brief analysis (in your README file) for each object file, indicating whether or not they passed your tests. If an object file didn't pass your tests include a description of each failure condition that must be addressed.
3. If, at any time, the code crashes then the problem is with your code. An object file *may* provide you with erroneous results but will never cause your program to fail outright. That is, how you're testing the object file is what is causing the crash and you need to adjust your tests accordingly.
4. The test framework from question 1 should form a good foundation upon which to build the

test framework for this question.

5. Bonus marks will be awarded for a *good* test framework making use of files to define the test suite. If attempted, include a note in your README file (along with how to run your tests) and include any test files required to perform your tests.