Lab 5 – ADTs and Unit Testing

# Part A – ADT Implementation

In this exercise we are going to build an ADT called “TwoSet” which is very similar to a traditional set except that instead of a single occurrence of each element we are allowed to have up to two copies of the same value in the data structure. For simplicity, you can assume that items in the set are just integers.

The class interface functions for our ADT are as follows:

* put(x): insert a new item into the TwoSet. If there are already two copies of the value x in the TwoSet, this operation will be ignored.
* remove(x): remove one copy of element x from the TwoSet. If there is no element with the value of x in your TwoSet the ADT will ignore the operation. This would remove only one copy of the element x if the TwoSet contained two values.
* has(x): return true or false depending on if the TwoSet contains at least one copy of a value.
* count(x): return the number of items in the TwoSet with the value x. This function should return 0, 1, or 2.
* size(): return the total number of elements in the TwoSet, including values that appear multiple times.
* distinct(): return the number of distinct values in the TwoSet.

In addition to the class functions write the following stand-alone functions (i.e. functions that are not part of the class). These functions return a new TwoSet object but must not modify the original TwoSets.

* union(ts1, ts2): Create a union between the TwoSet ts1 and the TwoSet ts2. Remember that you should never have more than 2 copies of each element after the operation. The function should return a new TwoSet object.
* intersect(ts1, ts2): Create an intersection between the TwoSet ts1 and TwoSet ts2. The result will be the values that are in common between both TwoSet objects. If ts1 has two copies of the element 15 and ts2 has only one copy, then the result will have only a single copy of 15.
* difference(ts1, ts2): Compute the difference between the two TwoSet objects. The new object will have all the values of ts1 that were not in ts2.

The implementation can take on any form (dictionaries, sets, hash tables, lists). How the implementation is done is entirely up to you.

Please write the time complexity for each of the functions as a comment in the code.

# Part B – Unit Test

Write a unit test suite to fully test your ADT created in Part A.