Arthur Liou
CS362
Due: 2/17/19

Midterm 1 – Q2

Prompt: You are given a library, container.o and the following .h file signatures:

- void add (int n, container* c); /* adds n to c */
- int get (int n, container* c); /* returns 1 if n is in c, 0 otherwise */
- int remove (int n, container* c); /* returns 1 if n was in c; after return n is not in c! */
- container* newContainer(); /* returns a new container if memory avail */
- void **removeAll**(int n, container* c); /* removes all instances of the number n from c*/.
- int size(container* c);/* returns the number of elements in c*/

1) Write **three** different **test cases** that you feel will adequately test and verify that **removeAll** properly deletes all copies of the number n from the container c. Please list the assertions you would make to verify your expected results in your unit test.

Response:

As part of writing three different test cases, I will write a positive test case, negative test case, and an edge / boundary case. The removeAll() function will remove all instances of the number n from container c.

I'm going to revise my vision / approach of this problem by viewing the container.o as an array.o type of file, as all the functions are relevant and would be present for an array, albeit with perhaps different method names. My examples / pseudocode are loosely based on C. They will not be compliable and only exist to serve as an idea of the process the test case will look like. Test cases as follows below.

Positive Test Case: This test case will prove that the removeAll() function will work if the number n is in c. As I'm looking for just one test case (out of the three), at minimum, the setup for this test case will be one container C that is already prepared with the number we want, then run the removeAll() function on the container.

Example / Pseudo Code where n = 7:

```
c = [1, 2, 3, 4, 5, 6, 6]; // Set up Container as c
    // This set up can be split off into more different parts, such as involving integration testing. This could include newContainer, multiple add(), check size() when the adds are done, then proceeding with the removeAll(). For this Example test case, I could also have 2+ adds of 7, so we can make sure that multiple n's / 7's are remove.d
removeAll(7, c)
assert(get(7, c) == false);
// End Test Case as 7 has been removed from the container.
```

Negative Test Case: This test case will prove that the removeAll() function will not do anything if the number n is not in c. As I'm looking for just one test case (out of the three), at minimum,

the setup for this test case will be one container C that is already prepared with the numbers we want and are not n, then run the removeAll() function on the container. We expect the function to “do” nothing, a get on n (7) will return false / 0, and the container size to be the same before and after.

Example / Pseudo Code where n = 7:

```
c = [1, 2, 3, 4, 5, 6, 6]; // Set up Container as c
size1 = size(C); // Grab the size of container, which is 7. size1 = 7
removeAll(7, c);
assert(get(7, c) == false);
Size2 = size(C); // Grab the size of container, which is 7. Size2 = 7
assert(size1 == size2);
// End Test Case;
```

Edge / Boundary Cases: This test case will prove that the removeAll() function will not do anything / exit early if C is not defined, does not exist, or is empty. For this style of data structure, one of the fundamental edge cases that should always be tested is the existence of the data structure and if it is populated.

```
assert(C); //Assert that C exists
assert(size(C) > 0); //assert that C contains at least one number
```