Name: Ethan Huang

ID: 406244368

**Notable Obstacles:**

The first two functions were relatively easy to implement, the main problem was figuring out how to do this in a loop. However, the last two were definitely more difficult. The combination function was especially difficult as it would normally be solved with a nested for-loop, so it took some thinking to come up with having two function calls in each function. The other challenge I faced was figuring out data types. I wasn’t aware that using square brackets with a string returns a character, and as a result my program was not appending characters correctly. I had to use substrings instead for a string output.

**Test cases:**

assert(modulo(0, 3) == 0);

* This tests if it works with 0

assert(modulo(1, 13) == 1);

* modulo 1 is also a edge case, as it should always return 1

assert(modulo(101, 10) == 1);

* Testing if the function is working

assert(occurrences(4000000, 0) == 6);

* Testing if function works with larger numbers

assert(lucky7s("abbba") == "ab77b77ba");

* Tests for multiple repeating characters in a row

assert(lucky7s(“aaaaaaa”) == “a77a77a77a77a77a77a”)

* Tests for if all letters are the same

assert(lucky7s(“bar”) == “bar”)

* Tests for no repeating characters

int other[5] = { 10, 8, 2, 6, 4 };

assert(combinations(other, 5, 10) == true);

* Tests for if no sums are required

assert(combinations(other, 5, 14) == true);

* Tests if sums work if there is a gap

assert(combinations(other, 5, 0) == false);

* Tests edge case where target=0

The only case that causes combinations to fail is if the target is equal to 0 and all of the numbers in the array are non-zero. This is because my combinations function tests by subtracting by target and checking if target==0. This doesn’t work if target is initially 0.