**Name: Yiming Xue Student ID: 806283777**

The first obstacle I encountered was when I wrote the function occurrence(), I tried to convert number into string so that every digit in the number can be split apart and turn into character. However, I checked the guideline and found out string function to\_string() is not allowed to use. So, I adjusted the method and obtained the digits by calculating the remainder when dividing the number by 10.

The second problem occurred when I implemented the last function combinations(). I initially considered creating an additional helper function. However, after I did the worksheet of week2 and checked the answers, I noticed that the last question is very similar to combinations(). As a result, I decided to employ a similar method.

These are test data:

// remaider = 0 or otherwise, dividends = 0.

assert(modulo(70, 7) == 0);

assert(modulo(9, 9) == 0);

assert(modulo(0, 11) == 0);

assert(modulo(15, 4) == 3);

assert(modulo(13, 2) == 1);

// test when d and numbers are both or either zero

// test normal occurrences count

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

assert(occurrences(0, 13) == 0);

assert(occurrences(1001, 0) == 2);

assert(occurrences(120, 0) == 1);

assert(occurrences(333333, 5) == 0);

assert(occurrences(333333, 3) == 6);

// test when no repetition or multiple repetitions

assert(lucky7s("abc") == "abc");

assert(lucky7s("Llo") == "Llo");

assert(lucky7s("cctv") == "c77ctv");

assert(lucky7s("lolll") == "lol77l77l");

assert(lucky7s("bbbb") == "b77b77b77b");

// test the adds-up will use once and case when they are partially or all needed

**int** a[5] = {1,2,3,11,13};

assert(combinations(a, 5, 7) == **false**);

assert(combinations(a, 5, 20) == **false**);

assert(combinations(a, 5, 50) == **false**);

assert(combinations(a, 5, 3) == **true**);

assert(combinations(a, 5, 17) == **true**);

assert(combinations(a, 5, 30) == **true**);