**Name**: Zihan Xia

**UCLA Id Number**:205838465

**A brief description of notable obstacles you overcame:**

Recursion is a more obscure and complex way of thinking, so it’s hard for me at first to think about the question in a different way than looping. For the modulo one, it’s simply hard for me to come up with the solution using subtraction every time with recursion, which I need time to digest. I think the discussion worksheet is helpful for understanding recursive logic. For the combination one, I didn’t come up with any ideas of how and where to start at the beginning, but the TA’s office hour helps a ton, which suggests me a smart way using || operation to deal with every single case in the function.

**A list of the test data that could be used to thoroughly test your functions, along with the reason for each test.**

**int main() {**

**using namespace std;**

**// test code**

**assert( modulo( 100, 10 ) == 0 );**

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

**assert( modulo( 12, 5 ) == 2 );**

**assert( modulo( 33, 6 ) == 3 );**

**//added**

**assert(modulo(10,10) == 0);**

**assert(modulo(10,100) == 10);**

**assert(modulo(300,20) == 0 );**

**assert(modulo(0, 7) == 0); // Testing with m = 0**

**assert(modulo(20, 1) == 0); // Testing with n = 1**

**assert(modulo(17, 17) == 0); // m equals n**

**assert(modulo(22, 7) == 1); // Typical case**

**assert(modulo(56, 9) == 2); // Typical case**

**assert(modulo(1000, 3) == 1); // Large values**

**// assert(modulo(123456, 789) == 579); // Large values**

**assert(modulo(7, 10) == 7); // m < n**

**assert(modulo(7, 7) == 0); // m = n**

**assert( occurrences( 567, 2 ) == 0 );**

**assert( occurrences( 128874, 2 ) == 1 );**

**assert( occurrences( 212121, 2 ) == 3 );**

**assert(occurrences(0, 0) == 0); // Edge case, both parameters are 0**

**assert(occurrences(0, 7) == 0); // Edge case, number is 0, d is not found**

**assert(occurrences(7, 0) == 0); // Edge case, d is 0, number is not found**

**assert(occurrences(123456, 7) == 0); // d not found in the number**

**assert(occurrences(777777, 7) == 6); // All digits are 7**

**assert(occurrences(987654321, 1) == 1); // Only one 1 in the number**

**assert(occurrences(123456789, 0) == 0); // No zeros in the number**

**assert(occurrences(123456789, 5) == 1); // Only one 5 in the number**

**assert(occurrences(0, 5) == 0); // Testing with number = 0**

**assert(occurrences(55555, 5) == 5); // All digits are 5**

**assert(occurrences(12345, 6) == 0); // No occurrence of 6**

**assert(occurrences(123450, 0) == 1); // Occurrence of 0**

**assert(occurrences(999999999, 9) == 9); // All digits are 9**

**assert(occurrences(123456789, 0) == 0); // No occurrence of 0**

**assert(occurrences(98764321, 5) == 0); // No occurrence of 5**

**assert(occurrences(112233, 1) == 2); // Repeated digits**

**assert(occurrences(7778778, 7) == 5); // Repeated digits**

**assert(occurrences(987643100, 5) == 0); // No occurrence of 5**

**assert( lucky7s( "cs32" ) == "cs32" );**

**assert( lucky7s( "Apple Baseball" ) == "Ap77ple Basebal77l" );**

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

**assert(lucky7s("nihao") == "nihao");**

**assert(lucky7s("nn") == "n77n");**

**assert(lucky7s("12345") == "12345"); // No repeating characters, the string remains the same**

**assert(lucky7s("aaabbbcccdddee") == "a77a77ab77b77bc77c77cd77d77de77e"); // Repeating characters**

**assert(lucky7s("z") == "z"); // Single character string**

**assert(lucky7s("") == ""); // Empty string**

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

**assert( combinations( array, 5, 10 ) == true );**

**assert( combinations( array, 5, 18 ) == true );**

**assert( combinations( array, 5, 15 ) == false );**

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

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

**assert( combinations( other, 5, 18 ) == true );**

**assert( combinations( other, 5, 15 ) == false );**

**//added by me**

**assert(combinations(other, 5, 3) == false);**

**assert(combinations(other, 5, 20) == true );**

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

**int array2[3 ] = {2,3,4};**

**assert( combinations( array2, 3, 2 ) == true );**

**assert( combinations( array2, 3, 3 ) == true );**

**assert( combinations( array2, 3, 4 ) == true );**

**assert( combinations( array2, 3, 5) == true );**

**assert( combinations( array2, 3, 6) == true );**

**assert( combinations( array2, 3, 7) == true );**

**assert( combinations( array2, 3, 9) == true );**

**assert( combinations( array2, 3, 12) == false );**

**assert( combinations( array2, 3, 16) == false );**

**int array3[4 ] = {2,3,4,100};**

**assert( combinations( array3, 4, 2 ) == true );**

**assert( combinations( array3, 4, 3 ) == true );**

**assert( combinations( array3, 4, 4 ) == true );**

**assert( combinations( array3, 4, 100) == true );**

**assert( combinations( array3, 4, 9) == true );**

**assert( combinations( array3, 4, 102) == true );**

**assert( combinations( array3, 4, 104) == true );**

**assert( combinations( array3, 4, 8) == false );**

**assert( combinations( array3, 4, 200) == false );**

**// Additional test cases**

**assert(combinations(other, 5, 3) == false);**

**assert(combinations(other, 5, 20) == true);**

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

**int array4[3] = {2, 3, 4};**

**assert(combinations(array4, 3, 2) == true);**

**assert(combinations(array4, 3, 3) == true);**

**assert(combinations(array4, 3, 4) == true);**

**assert(combinations(array4, 3, 5) == true);**

**assert(combinations(array4, 3, 6) == true);**

**assert(combinations(array4, 3, 7) == true);**

**assert(combinations(array4, 3, 9) == true);**

**assert(combinations(array4, 3, 12) == false);**

**assert(combinations(array4, 3, 16) == false);**

**int array5[4] = {2, 3, 4, 100};**

**assert(combinations(array5, 4, 2) == true);**

**assert(combinations(array5, 4, 3) == true);**

**assert(combinations(array5, 4, 4) == true);**

**assert(combinations(array5, 4, 100) == true);**

**assert(combinations(array5, 4, 9) == true);**

**assert(combinations(array5, 4, 102) == true);**

**assert(combinations(array5, 4, 104) == true);**

**assert(combinations(array5, 4, 8) == false);**

**assert(combinations(array5, 4, 14) == false);**

**assert(combinations(array5, 4, 200) == false);**

**cout << "all tests passed!" << endl;**

**return( 0 );**

**}**