Anya Smolentseva 305503018

1. Implementing the first function was extremely difficult for me. In order to understand it better, I broke it down into multiple functions, which tested different aspects of the string. For instance, one function checked whether the string started with R, another function checked if it had leading zeros, etc. Of these tiny functions, I had a difficult time checking for leading zeros. While the provided extractNumber function worked for leading zeros, it returned false for when the positive or negative test case was just 0. To fix this, I decided to not use the extractNumber function and instead I used a for loop that checked if the character was a 0 and the next character was a digit, then to return false. I also had a difficult time implementing the function that would return multiple batches as valid. To fix this, I compensated by adding more details to my other functions so that while my program didn’t treat each individual batch as an individual substring, it would continue looping until the end and would still find errors in any part of the string (like leading zeros) to be invalid.
2. I designed my program by making multiple functions for the isValidResultString function.
   1. noSpacesInString, checks that there are no spaces and that the length of the string has at least length 6.
   2. correctCharacters, checks that the string only has ‘R’ ‘+’ ‘-’ and digits.
   3. startsWithR, checks that string starts with ‘R’
   4. extractNumber, turns numbers in a string to integer values
   5. plusAndNegative, checks that there are an equal amount of ‘+’ and ‘-’, and that for each batch there is one ‘+’ and one ‘-’
   6. posNextToNeg, checks that there aren’t two ‘+’ in one batch or 2 ‘-’ in one batch
   7. firstZero, checks that there isn’t a 0 after ‘R’
   8. leadingZeros, checks that there aren’t leading zeros
   9. equalsTotal, checks that positive and negative cases equal total cases
   10. numberBatches, validates that a string can have multiple Rs

positiveTests

1. One for loop, if character is ‘+’ go to next index and extract that number (assuming the string is valid) to find value of positive tests

negativeTests

1. One for loop, if character is ‘-’ go to next index and extract that number (assuming the string is valid) to find value of negative tests

totalTests

1. One for loop, if character is ‘R’ go to next index and extract that number (assuming the string is valid) to find value of total tests

batches

1. One for loop with an rcount, everytime character is ‘R’, add one to rcount, and return rcount (assuming string is valid) to find number of batches

3)

Testing for spaces/correct length:

   assert(isValidResultString("") == false);

    assert(isValidResultString("    ") == false);

    assert(isValidResultString("1R") == false);

assert(isValidResultString("R5") == false);

Testing for batch that starts with 0:

    assert(isValidResultString("R0+1-1") == false);

Testing that ‘+’ and ‘-’ aren’t next to each other

    assert(isValidResultString("R5+-1") == false);

Testing that values add up

    assert(isValidResultString("R2+2-0") == true);

assert(isValidResultString("R5-2+3") == true);

assert(isValidResultString("R50+30-25") == false);

Testing for leading zeros

    assert(isValidResultString("R5+00003-00002") == false);

Testing that values in the 10s place add up

    assert(isValidResultString("R50+30-20") == true);

    Testing that multiple batches work

    assert(isValidResultString("R50+30-20R10+5-5R2+1-1") == true);

Testing that incomplete batches/incorrect characters don’t work

    assert(isValidResultString("R5+3-2R") == false);

    assert(isValidResultString("R5+3-2h") == false);

Testing positive tests

    assert(positiveTests("R2+1-1") == 1);

    assert(positiveTests("R5+2-3R5+2-6") == -1);

assert(positiveTests("R10-7+3R0+0-0") == -1);

Testing negative tests

    assert(negativeTests("R2+1-1") == 1);

 assert(negativeTests("R5+3-2R20+0-20") == 22);

    assert(negativeTests("R5+6-1R3+2-1") == -1);

assert(negativeTests("R200+150-50R5+-32") == -1);

Testing batches with only ‘+’ or only ‘-’

    assert(isValidResultString("R5+3+2R1-1-0") == false);

Testing the total number of batches

    assert(batches("R2+1-1R3+2-1") == 2);

assert(batches("R2+2-0") == 1);

assert(batches("R5+0001-00004") == -1);

assert(batches("R100+-100") == -1);

Testing total tests

    assert(totalTests("R100+50-50R10+5-5") == 110);

 assert(totalTests("R2+1-1") == 2);

assert(totalTests("R5+3-3R2-1+4") == -1);

 assert(totalTests("R1+1-0R2+2-0R1+1-0") == 4);

 Doesn’t work // assert(positiveTests("R3+2-0R1+1-1") == -1);

This type of test case doesn’t work because I didn’t have time to figure out how to treat each batch as a singular string. While the total cases do add up correctly, individually in each batch, the values do not add up to the total tests in the batch.