Duncan Di Mauro

SID: 805163177

February 2, 2021

Report 3

Obstacles:

This project was much more difficult that Project 2 and required me to overcome various obstacles. The most significant obstacle that I had to overcome was figuring out how to approach this project and the isValidResultsString function. I followed the advice in the “Project 3 FAQ” and began by checking if the first character of a string was ‘R’. Then, I continued onto the next characters, making sure they were numbers, followed by either a + or – sign, followed by numbers, etc. During this process, it was very difficult to figure out how to check for an undefined number of digits after the characters ‘R’, ‘+’, and ‘-‘. For example. if I knew that there would only be 2 digits after R and 1 digit after ‘+’, I could simply write code making sure that those specific characters were digits. However, I needed to allow for an endless number of digits to be checked. In order to achieve this, I wrote a function called CalculateNumberInString which takes a string and a number as an input. The input number represents the location of the first character in a number that is within a string. The function includes a while loop that says, “while the character in the string is a digit, extract that character and add it into a new string.” Once the while loop ends, the function returns a string of numbers. For example, “142.” I use this function in all my other functions. In order to change the string to an int type, I used the command stoi. Other challenges in this project included understanding how variables work within functions and loops and the necessity to pre-allocate values to these variables under certain circumstances. Once I was able to write my isValidResultString function, I had no problem writing the other functions.

Description of Program Design:

**string CalculateNumberInString:** In order to write all of my functions, I created one other function called CalculateNumberInString, which extracts a number that exists within a string, using the string as an input and a number (int) which is equal to the location of the first digit within the number. For example, if the string is “abc152def”, then the function would take string number = “abc152def” and a variable int k = 3, since number[3] = ‘1’ which is the first digit of the number we are trying to extract. The function includes a while loop that says, “while number[k] is a digit, extract that character and add it into a new string. Then, do k += 1 and go back to the beginning of the loop.” Once the while loop ends, the function would output “152.”

**bool isValidResultString:** This function performs a series of checks on a given string and outputs false if the string fails any of the checks. If a string passes all the checks, then the function outputs true. The function starts out by using an if statement to check if the first character in a results string is the letter ‘R.’ Then, the function initializes all variables that will be used in the rest of the function. A while loop is then created which states while (k < results.size()), where k is initialized as 0. In this loop, k represents the location of a character in the results string (i.e. results[k]). The loop performs a series of checks until the string is over. The first check is whether the string begins with the letter ‘R.’ After this check, 1 is added to k. Then, the string checks if the next character is a digit and is also not equal to 0. Then 1 is added to k. If this check is passed, the function CalculateNumberInString is called. The function uses stoi to extract an int value from this string, which equals the number of tests in one batch. Then, the length of the string is added to k in order to move k to the position of the character located after the number. The same principals which check for ‘R’ and extract the number which follows are then applied to the + sign, - sign, and numbers that follow them. If the number extracted after the positive sign and the number extracted after the negative sign do not add up to the number of tests in a batch, the function outputs false. Additionally, if the batch does not include exactly one minus sign and one plus sign, the function returns false. After all these checks are performed and a value is added to k which moves it to the next character, the loop restarts, with the first check being if results[k] != ‘R’. After all batches have been checked, the string comes to an end, or the string fails one of the checks, the function ends and outputs either “true” or “false” depending on what happened. This function, isValidResultString, is used in the beginning of all the following functions. If the string is not valid, all of the following functions just return -1.

**int positiveTests and int negativeTests:** These two functions are the exact same. Their only difference is the output statement at the very end of the program. These functions go through the exact same process as isValidResultString, except they do not perform any checks, since isValidResultString is called at the beginning of the program. Instead, they store the values extracted after the + sign and – sign into variables. In each iteration of the while loop, these variables are updated to add the values extracted from the previous batches. The values after the + sign and the values after the – sign are easily separated using if statements.

**int totalTests:** This function is essentially the same as positiveTests and negativeTests, however it stores and updates the value found after the ‘R’ in each batch. After the function ends, it returns the number of total tests.

**int batches:** This function is the same as totalTests, however instead of calculating the number of total tests, it simply calculates the number of batches. In this function, the number of batches is equal to the amount of iterations of the while loop.

**int main():** This is used to check the functionality of all the other functions. It simply asks the user to type in a results string and applies it to all of the functions. Then, the output of each function is printed, and the results can be seen.

List of Test Data:

Many of the strings provided by the professor in the Project 3 description can be used to test my program (I have also included many additional test strings). Since my functions each provide a series of checks, the test data should include strings which check for each test and make sure the program functions as intended.

r1+0-1                              (batch must be reported with R)

R1+-1                               (a number of positive tests is required)

R1+1-                               (a number of negative tests is required)

R1+0-1   asdfR               (extra characters not allowed)

R5+00003-0002          (leading zeros not allowed)

R5+0-0              (positive and negative results must equal the total number of tests)

R0+0-0                            (must have a number of cases to report)

R0-0+0                            (must have a number of cases to report)

"R3-1+2        "  (No other characters allowed. The quotes show that spaces are not valid)

R2-1+1R0+0-0              (One valid batch and one invalid batch which has zero cases reported)

R5-3-2R2+2+0              (a + and a - are required inside every batch of results)

R1+0-1 abcdefg             (no extra characters allowed)

R1abcdefg+0-1              (no extra characters allowed)

R1+0-1R1-0+1               (two batches of results, two total tests being reported, one being

positive, one being negative)

R5-2+3                            (Valid results string with only one batch and only single digit numbers)

R1+1-0R1-1+0R1-1+0R1+1-0    (valid string with multiple batches)

R360-300+60                (valid string - int values are not limited to a single digit)

R2400-1200+1200 (valid string - int values are not limited to a single digit)

R2400-R1200 (invalid string – must have positive and negative results in batches)

R50+20- (invalid string – must have a number of negative results in a batch)

R50+20-30 R15-8+8 (invalid string – must not have any spaces or invalid characters)

R300+150+150 (string must have both positive and negative tests)

R300 +150-150 (string must not have any spaces)

R 300+150-150 (string must not have any spaces)