Junho Choi

Prof. Smallberg

CS 31

4 May 2020

Project 3 Report

During the coding of this project, I ran into numerous obstacles. I began coding the hasValidSyntax function first. Though getting it to take in an order string with one state (eg CA325+, nY42-) and returning the correct boolean value was pretty straightforward, I began facing problems when trying to put multiple order strings together. I was able to overcome this issue by integrating an infinite loop that would check for the order string and having an index counter that moves through the order string. When I started programming the countCases function, I quickly ran into multiple obstacles. One of them was casting the character values of the order string into an integer value so I can count the number of cases for that status. I was able to overcome this by making a function named stringToInt, which would take in a string value and return the corresponding integer value. Another issue that I ran into was returning 2 if any of the orders was 0. I fixed this by adding a function called isZeroOrders which goes through the order string and checks if there are any orders regardless of status with a value of 0.

My logic for hasValidSyntax is pretty straightforward; I checked the state code, the number, then the status, and repeated this until I found an error or reached the end of the order string without errors. Here is the pseudocode :

hasValidSyntax :

...

repeatedly :

check to make sure next two index is a valid state code

if not valid state code,

return false

check how many numbers there are

if no numbers,

return false

check if status is + or -

if not + or -,

return false

check if end of string

if end of string,

return true

repeat for next order code

...

My logic for countCases is a lot more complex. Though the conditions for returning 1 (checking if hasValidSyntax returns true or false) or 3 (checking if parameter status is == to + or -) was straightforward, returning 0 or 2 was more complex. Here is the pseudocode for returning 0 and 2 :

isZeroOrders (returning 2) :

…

repeatedly until reaching the order string :

skip the state code (the order string being syntactically correct is a prerequisite)

convert the number in the string to an integer value up to before the status symbol

if the value is 0,

return false (return 2 in countCases)

skip status symbol, and begin next order code

if it goes through the order string without error, return true

…

countCases (returning 0) :

…

repeatedly until reaching the end of the order string :

skip the state code

convert the number in the string to an integer value up to before the status symbol

if the status code after the number matches the status,

add the value to total count

continue to next order code

return 0

…

Test Cases :

**hasValidSyntax :**

hasValidSyntax(“cA230+Ny33-WY23+gU42-”);

hasValidSyntax(“aZ30+IL34-wY23-vA3435-”);

hasValidSyntax(“fL46-kS34-pR3573+dc674+”);

hasValidSyntax(“”);

tests if function correctly identifies correct order strings

hasValidSyntax(“cA230Ny53-WY2353+gU942-”);

tests if function can detect missing status symbol

hasValidSyntax(“fL350+xP34-pR873+dc34+”);

hasValidSyntax(“nY30+IL34+wY23vA3435-”);

hasValidSyntax(“d30+wA343-”);

hasValidSyntax(“shd78+wA343-”);

tests if function can detect nonexistent state code

hasValidSyntax(“fL34+ pr343-”);

hasValidSyntax(“fL34+p r343-”);

tests if function can detect incorrect spaces

hasValidSyntax(“Ca83+mt23--”);

hasValidSyntax(“aB83-+nY23--”);

tests if function can detect incorrect incorrect number of/misplaced status symbols

hasValidSyntax(“4ca83+4mt2-”);

hasValidSyntax(“aB83-nY23-4”);

tests if function can detect misplaced numbers

hasValidSyntax(“ “);

hasValidSyntax(“ “);

tests if function can detect only spaces

**countCases : (doesn’t include tests that would have already failed hasValidSyntax)**

countCases(“cA230+Ny53-WY2353+gU942-”, ‘-’, x);

countCases(“aZ30+IL34-wY23-vA3435-”, ‘-’, x);

countCases(“lA30+ok34-TN23-oH5-”, ‘+’, x);

tests if function correctly returns number of cases

countCases(“aZ30+IL34-wY23-vA3435-”, ‘$’, x);

countCases(“lA30+ok34-TN23-oH5-”, ‘%’, x);

countCases(“lA30+tn34-or23-oH5-”, ‘)’, x);

tests if function correctly recognizes incorrect status symbol

countCases(“aZ0+IL34-wY23-vA3435-”, ‘+’, x);

countCases(“lA30+ok0-TN0-oH5-”, ‘-’, x);

countCases(“ak30+ok34-TN23-gu0-”, ‘+’, x);

tests if function correctly recognizes zero orders