CS 31 Lecture 1 – Professor Smallberg Leo Gretzinger

Project 3 Report 10/28/18

**Obstacles:**

The main source of obstacles that I experienced within this project was creating my isValidStateForecast function. This function involves a large amount of while loops and if statements that attempt to account for every individual case of invalid and valid state forecasts. As a result, to create this function, I traced through a lot of different code by hand, debugging and trying different inputs and indexes and ultimately ended up creating a complex web of if statements that accounts for every case of input. The second, less challenging obstacle involved creating the hasProperSyntax method. The main problem within this process was figuring out how to analyze each state forecast with my isValidStateForecast function. I settled two sets of while loops: one for all the state forecasts besides the last one, and one for the last one.

**Pseudocode:**

isValidUppercaseStateCode function

variable with all valid state codes

return whether the code has a length of two, has periods and matches a code from the variable

isValidStateForecast function

return true if empty string

if the string is longer than one character

test if the state code is valid, using the above function

until the location is at the end of the input string:

if this character is a digit and it’s not the end of the string, continue

test whether the next character is a digit or letter and if it’s the end of the string – if it is a digit then a letter, or if it is a letter and it continues or if it is the end of the string, the syntax is valid

return whether the individual state forecast is valid or not

hasProperSyntax function

if the string is longer than one character and the last character is not a comma:

count the number of forecasts in the string

separate all the forecasts into separate strings using a temporary string and assess whether they are valid

return whether the entire poll string data has valid syntax or not

tallySeats function

test if the poll data string has invalid syntax or if the party code is not a letter

cycle through the entire string and if a character matches the party code add the number (and the number before that if applicable) to the tally of seats

**Test cases:**

My main function assertion cases are:

**//Tests if empty string is valid**

assert(hasProperSyntax(""));

**//Tests if a number as the first character is valid**

assert(!hasProperSyntax("2"));

**//Tests if a lowercase state code is valid**

assert(hasProperSyntax("ny"));

**//Tests if a state code, digit, then letter is valid**

assert(hasProperSyntax("ny3d"));

**//Tests if a missing party code isn’t valid**

assert(!hasProperSyntax("ny2"));

**//Tests if a forecast with multiple parties is valid**

assert(hasProperSyntax("ny3r4d"));

**//Tests if multiple forecasts are valid**

assert(hasProperSyntax("ny3r34f,ga34f6d,ct6y5t,az5g66h"));

**//Tests if a more than two party codes in a forecast is valid**

assert(hasProperSyntax("ny33r34f54r23f67j4h88j3f1d,ga34f6d,ct6y5t,az5g66h"));

**//Tests if two digits then one digit before the party codes in a forecast are valid**

assert(hasProperSyntax("ny33r4f,ga34f6d,ct6y5t,az5g66h"));

**//Tests if one digit then one digit before the party codes in a forecast are valid**

assert(hasProperSyntax("ny3r4f,ga34f6d,ct6y5t,az5g66h"));

**//Tests if a missing party code with multiple forecasts is invalid**

assert(!hasProperSyntax("ny3r4,ga34f6d,ct6y5t,az5g66h"));

**//Tests if a comma at the end of the poll data string is invalid**

assert(!hasProperSyntax("ny3r4d,ga34f6d,ct6y5t,az5g66h,"));

**//Tests if multiple forecasts with multiple party codes are valid**

assert(hasProperSyntax("CT5D,NY9R17D1I,VT,ne3r00D"));

**//Tests if in an invalid state code creates an invalid string**

assert(!hasProperSyntax("ZT5D,NY9R17D1I,VT,ne3r00D"));

int seats;

**//Tests if a valid poll data string and party code creates the correct number of seats**

seats = -999;

assert(tallySeats("CT5D,NY9R17D1I,VT,ne3r00D", 'd', seats) == 0 && seats == 22);

**//Tests if an invalid party code returns the correct error number**

seats = -999;

assert(tallySeats("CT5D,NY9R17D1I,VT,ne3r00D", '%', seats) == 2 && seats == -999);

**//Tests if an empty string is valid in this function**

assert(tallySeats("", 'd', seats) == 0 && seats == 0);

**//Tests if an invalid poll data string returns the correct error number**

seats = -999;

assert(tallySeats("2", 'd', seats) == 1 && seats == -999);

**//Tests if only the state code is valid**

assert(tallySeats("ny", 'd', seats) == 0 && seats == 0);

**//Tests if the tally of seats is correct with a valid input**

assert(tallySeats("ny3r4d", 'd', seats) == 0 & seats == 4);

**//Tests if the tally of seats is correct with an input of multiple forecasts**

assert(tallySeats("ny3r34f,ga34f6d,ct6y5t,az5g66f", 'f', seats) == 0 && seats == 134);

**//Tests if the tally of seats is correct with an input of multiple long forecasts**

assert(tallySeats("ny33r34f54r23f67j4h88j3f1d,ga34f6d,ct6y5t,az5g66h", 'j', seats) == 0 && seats == 155);

**//Tests if the tally of seats is correct with different party codes**

assert(tallySeats("ny33r4f,ga34f6d,ct6y5t,az5g66h", 'f', seats) == 0 && seats == 38);

**//Tests if the poll data string is still valid with a one digit and one digit layout**

assert(tallySeats("ny3r4f,ga34f6h,ct6y5t,az5g66h", 'h', seats) == 0 && seats == 72);

**//Tests if a missing party code in a forecast returns the correct error number**

seats = -999;

assert(tallySeats("ny3y4,ga34f6d,ct6y5t,az5g66h", 'd', seats) == 1 && seats == -999);

**//Tests if a comma at the end of a poll data string returns the correct error number**

seats = -999;

assert(tallySeats("ny3r4d,ga34f6d,ct6y5t,az5g66h,", 'd', seats) == 1 && seats == -999);

**//Tests if an invalid party code returns the correct error number**

seats = -999;

assert(tallySeats("CT5D,NY9R17D1I,VT,ne3r00D", '\*', seats) == 2 && seats == -999);

**//Tests if a integer character party code returns the correct error number**

seats = -999;

assert(tallySeats("CT5D,NY9R17D1I,VT,ne3r00D", '0', seats) == 2 && seats == -999);

cout << "All tests succeeded" << endl;