Angela Kan

Project 3 Report

1. **Obstacles:** I had one rather notable obstacle while writing the isSyntacticallyCorrect function: I wasn’t quite able to correctly iterate through the string the right amount of times for each section of the state forecast, and so kept getting failed assertions. After tedious tracing, I finally realized I was iterating through too many characters in the string at once, and fixed the code, but this took me a great deal of time to finalize. I also got stuck checking for index values that were out of bounds, but worked this out with the use of multiple “if” statements.
2. **Description:** Check poll data string for correct syntax, then tally up votes for a given party and string:

*Check string for correct syntax:*

*Check for empty string*

*Go through each character of the string:*

*Check for digits at beginning*

*return false if incorrect format*

*Check for correct state code*

*return false if incorrect format*

*Check for letter party character*

*Move forward in string if present*

*Tally votes based on party char and string data:*

*if string isn’t syntactically correct*

*return 1*

*if party char isn’t a letter*

*return 2*

*Otherwise convert string to all uppercase*

*convert party char to uppercase*

*while poll data string isn’t empty:*

*if two digit number*

*convert into integer*

*if one digit character*

*convert into integer*

*move forward through string*

*check for instances of zero electoral votes:*

*return 3 if yes*

*otherwise if party code matches string:*

*add number to votes for that party*

*clip the state forecast that was just reviewed*

*passes number of votes into reference parameter*

*Check if state code is a valid uppercase one (provided code)*

*check if state code is correct:*

*checks for numbers after digit:*

*adds to number of votes*

*moves to next letter position*

*Check for 2+ digit numbers*

*return false*

*if reaches end of string:*

*stop iterating, return false*

*change all char of poll string into uppercase*

*iterate through state code*

*add state code to new string*

*if reaches end of string:*

*return false*

*check for validity of uppercase state code*

*return true if yes*

*if not, return false*

1. **Test Cases:**

Empty String: assert(isSyntacticallyCorrect(""));

One digit state forecast: assert(isSyntacticallyCorrect("5CAD"));

Two digit state forecast: assert(isSyntacticallyCorrect("55CAD"));

Missing char in state code: assert(!isSyntacticallyCorrect("55C"));

Missing state code (2 digit number): assert(!isSyntacticallyCorrect("55"));

Missing state code (1 digit number): assert(!isSyntacticallyCorrect("5"));

State forecast w/ 2 zeros: assert(isSyntacticallyCorrect("00CAD"));

State forecast w/ 1 zero: assert(isSyntacticallyCorrect("0cAd"));

Non-digit first character: assert(!isSyntacticallyCorrect("!"));

Incorrect party char type: assert(!isSyntacticallyCorrect("4ca0"));

Too many digits for electoral vote: assert(!isSyntacticallyCorrect("433cad"));

Too many char for party code: assert(!isSyntacticallyCorrect("43cadd"));

Wrong character type+multiple characters for party code: assert(!isSyntacticallyCorrect("43ca4d"));

Correct poll data string with multiple state forecasts: assert(isSyntacticallyCorrect("38TXR55CAD6Msr29nYd06UTL"));

Unsupported character in poll data string: assert(!isSyntacticallyCorrect("38TXR:55CAD"));

space in poll data string: assert(!isSyntacticallyCorrect("38TXR 55CAD"));

int votes = -999;

Correct case, checking for correct vote tally: assert(tallyVotes("38TXR55CAD6Msr29nYd06UTL", 'd', votes) == 0 && votes == 84);

votes = -999;

Detecting whether vote tally sets votes when it shouldn’t: assert(tallyVotes("38TXR55CAD", '%', votes) == 2 && votes == -999);

votes = -999;

Incorrect syntax for poll string: assert(tallyVotes("55C", 'R', votes) == 1 && votes == -999);

Multiple errors (syntax and party char): assert(tallyVotes("55C", '%', votes) == 1 && votes == -999);

Empty string and incorrect party char: assert(tallyVotes("", '@', votes) == 2 && votes == -999);

Empty string, correct party type: assert(tallyVotes("", 'R', votes) == 0 && votes == 0);

Correct case, checking case sensitivity of party: assert(tallyVotes("38TXR", 'r', votes) == 0 && votes == 38);

votes = -999;

Checking for return 3 (0 electoral votes, matching party): assert(tallyVotes("00TXR", 'r', votes) == 3 && votes == -999);

checking for return 3 (1 digit, still zero electoral votes): assert(tallyVotes("0TXR", 'r', votes) == 3 && votes == -999);

Different party type, zero electoral votes: assert(tallyVotes("00TXR", 'X', votes) == 3 && votes == -999);

Zero electoral votes for one state in a long poll data string: assert(tallyVotes("38TXR55CAD6Msr0nYd06UTL", 'd', votes) == 3 && votes == -999);

Correct string, with a number that has a zero in it: assert(tallyVotes("38TXR55CAD6Msr4nYd06UTL", 'l', votes) == 0 && votes == 6);