Jason Tran

CS31 Project 3

10/26/22

A) The most challenging obstacle I had to overcome was figuring out how to code the isValidPollString functions. There were a lot of rules that made it a valid one and more that made it an invalid one. A string with nothing in it was even a valid poll string. After trying to come up with a solution in only one function, I found out that it was easier to make multiple functions to check the pollstring. So basically, I split the string up to the state forecasts and that into party results. If one returned false then it was assumed that the whole string was invalid and therefore isValidPollString would return false.

Another obstacle was making sure I wouldn’t go out of range of a string index, when searching through the poll string. Basically, my program would search for the party letter and check if the character before and after is a digit, which would make it the party. However, if the character I was searching for was at 0, then subtracting one from that would be invalid. To fix this I started searching at position 2 since we know that the 1st two characters will be letters (state code).

B) My program consists 4 of my created functions: isValidPollString, isValidStateForecast, isValidPartyResutl, and countSeats. Basically, the poll string gets split up into individual stateforecasts, and the stateforecasts gets splits into the state code and party results, to each be tested in their respective function.

In Pseudocode:

*IsValidPollString:*

Uppercase all letters in string

For each character in string:

If letter or digit

Add to empty string

If comma

Check if valid stateforecast

Reset to empty string

Else

Return false

*isValidStateForecast:*

Check if first two characters is valid state code

For each character starting from position 2:

If digit or letter:

Add to empty string

else:

Return false

Check empty string through isValidPartyResult

*isValidPartyResult:*

For each character:

If a letter:

Add to party result count

For each character

If a digit:

Add to empty string

If a letter:

Add to the empty string

Check whether size of string is 2-3 with correct amount of digits:

If true:

Add to check count

Else:

Return false

Reset the empty string back to empty

Compare check count with party result count

*countSeats:*

Check if pollstring and party character is valid

For each character:

If a digit:

Add to an empty string

If party character letter:

Check if actual party letter or something else

Convert the empty string to integer

Add to seat count

Reset the empty string

If character:

Reset the empty string

C) Testing the isValidPollString function:

* “NY9R16D1I,VT,NJ3D5R4D,KS4R” test what a valid, possible poll-string could be.
* “KS4R, NV%D1R” testing what happens if there are characters other than a letter, comma, or digit.
* “” test what happens if it is an empty string.
* “NY9R16D1I,,VT,NJ3D5R4D,KS4R” test if two commas seperating each forecast would return false.
* “NY” testing if there are no party results.
* “NY1R1R1R11R” testing if a state forecast with the same party, with multiple party results would be true.
* “XX9R16D,CA3D5R” testing what happens if there is a wrong state code.
* “nY9r16D1i,Vt,NJ3D5r” testing with a mixture of lower cases and upper cases.

Testing the countSeats function:

* (“NY9R16D1I,VT,NJ3D5R4D,KS4R”, ‘r’, seats) testing if seats changes to the correct amount of seats the party, r, won and works with either lowercase party or uppercase party.
* (“XX9R16D,CA3D5R”, ‘d’, seats) testing if the function returns 1 since poll string is invalid (state code).
* (“,NY9R16D1I,VT,NJ3D5R4D,KS4R”,‘r’, seats) testing if the function returns 1 since poll string is invalid (random comma)
* (“NY9R16D1I,VT,NJ3D5R4D,KS4R”, ‘%’, seats) testing if the function returns 2, since party code is invalid.
* (“XX9R16D,CA3D5R” , ‘%’, seats) testing if the function returns either a 1 or a 2 since both are wrong.
* (“NY1r1r1r1r11r”, ‘r’, seats) testing if there are multiple party results from the same party works.