## Obstacles Encountered

* Before starting I was confused about how I could count votes for every party. I hadn’t realized that the spec had said that it would specify which party (with the party letter) I would need to count. I did not need to count the votes for every party.
* One problem I encountered with the while loop was that if the pollData string had a few leftover characters at the end my program said it was still valid(for example “R40TX” is valid but “R40TXo” shouldn’t be). To solve this I simply updated the strSize<4 condition to be strSize<4 && strSize>0. This kept any random non-predictions from being ignored by my program’s logic.
* I also forgot about the function included in the spec to check if a state code was valid or not so I was confused about how to verify the state codes. However, a friend (Rohan Soni) reminded me about the function provided and I proceeded to use it.
* I used Professor Smallberg’s tester to test my program. My program failed one test case. I was initially confused but realized that I had to set the voteCount variable to zero somewhere in computeVotes to “erase” it of any previous non-zero value it might’ve had

## Description of Program

### int stringtoint(string str, bool twodigit){ //custom function

Initialiize *num* to hold number from string

if (twodigit number){

rtr[0]\*10 becomes tens place of num

rtr[1] becomes ones place of num

return num

}

### isValidUppercaseStateCode(string stateCode){

checks if state code is valid by checking it against all possible state codes

}

### hasRightSyntax(string pollData)

* + For loop iterating through all characters in string pollData
    - Capitalize letter
    - Check if character in the string is alphanumeric (if char is not alphanumeric string is invalid)
  + While loop (as long as pollData string size>=4)
    - Our checks:
      * If 1st char isn’t a letter, state prediction is invalid *(party char doesn’t exist)*
      * If 2nd char isn’t a digit, state prediction is invalid *(vote number doesn’t exist)*
      * If 3rd char is a letter and the 4th char is not also a letter the *(this means state code is incomplete)*
      * If the 3rd char is a digit and the 4th and 5th chars are not letters the prediction is invalid *(state code is incomplete or not present)*
      * Take a substring of the two letters at the end of the prediction *(the state code)* and run it through the valid State checking function provided by Professor Smallberg
    - After checks for string validity, we remove 4 or 5 char long state prediction analyzed from the string by with str.replace(0,5,“”) or str.replace(0,4,“”)
    - strSize = size of new updated string;
    - Check if strSize<4
      * If yes, there are random extraneous characters. Conversely, if there are >=4 characters it is possible another valid prediction string exists
    - As long as strSize>=4 the while loop runs and continues removing 4 or 5 chars at a time from itself

### computeVotes(string pollData, char party, int& votecount)

#### Error checks:

* Call hasRightSyntax function() to check if pollData string is valid
* Check if pollData string has a 0 in it with for loop)
  + If zero is alone *(i+1 and i-1 are not digits)* → return 2
  + If 0 is coupled with another 0 *(i+1 or i-1 is 0)* → return 2
  + The zero is connected to another integer *(i+1 or i-1 is non-zero digit)* → program continues
* Check if party char is a valid letter with the isalpha() function

voteCount=0; *//erases any nonzero previous value voteCount may have had*

#### While loop (as long as strSize>=4):

* if (first char in pollData == party char)
  + Identify if state prediction is a 4 or 5 character long string.
  + Use stringtoint() to extract vote number from pollData string
  + Votecount += vote number from pollData string
* Remove 4 or 5 characters (depending on whether vote number was 1 or 2 digit number)
* Update strSize variable for the next iteration

## hasRightSyntax Tests

* Valid Case Tests:
  + R40TX (does it work for a single state prediction?)
  + “” (empty string test)
  + R40TXD54CAr6Msd28nYL06UT (does it work for strings with multiple predictions)
* Extraneous Character tests:
  + R40TXo (does it work for strings with extraneous characters at the end?)
  + R40TXoooo (edge case: if there are four extraneous characters)
  + R40TXooo (edge case: if there are three extraneous characters)
* Invalid character tests:
  + R:40TX (does it work for strings with random characters in the middle?)
  + "R40TXD54CAr6;Msd28nYL06UT "(random semicolon in middle and a space at the end)
  + "R40TXD54CAr6Msd28nYL06UT "(space at the end)
  + “ R40TXD54CAr6Msd28nYL06UT” (space at the start)
* Invalid vote number tests:
  + R00TX (does the function return 2 for an input of 0)
  + R0TX (does the function return 2 for an input of 0)
  + R000TX (does the function fail for three digit vote number input)
* Invalid string tests
  + R40TXD54CAr6Msd28nYL06U (without the T at the end)
  + R40TXD54CAr6Msd28nY06UT (without the L in the middle)
* State code tests
  + R40TXD54CAr6Msd28nYL06AA (AA is not a valid state)
  + R40TXD54CAr6Msd28nYL06Tx (Tx is a valid state)
  + R40TXD54CAr6Msd28nYL06tx (tx is a valid state)

## computeVotes Tests

*Note that we do not need to retest every string type because they have all been tested in hasRightSyntax tests already. We only need to test a few to verify that the logic remains correct in computeVotes. We do however still need to test every branch of computeVotes*

Simple Valid tests:

* (R40TX, r, 0): Does this output correct votes?
* (R40TX, r, 40): Does this output correct votes (does it turn 40 into 0 before adding on new votes?)
* (R4TX, r, 0): Works for single digit vote number in state prediction
* (D3CA, d, 4): Works for letters other than r

VoteCount unchanged tests:

* (R40TX, 3, 4): Test to see if voteCount remains unchanged since party char is invalid
* (Rds, 3, 4): Test to see if voteCount remains unchanged since string and party char are invalid
* (Rds, r, 4): Test to see if voteCount remains unchanged since string is invalid. Also to see if computeVotes returns 1.

Party char tests:

* (R40TX, R, 0): Test to see if party char uppercase works
* (R40TX, r, 0): Test to see if party char lowercase works
* (R40TX, 3, 0): Test to see if invalid party char is detected. computeVotes should return 3

0 electoral votes tests:

* (R00TX, r, 0): Does computeVotes return 2 for electoral votes of 0
* (R0TX, r, 0): Does computeVotes return 2 for electoral votes of 0

Other tests:

* (r40TX00, r, 0): Test to see if extraneous characters make the string invalid and computeVotes returns 1