One of the main obstacles I overcame was finding a way to consistently check what part of the code I was looking for, be it the party code, number of votes, state code, etc. I ended up doing this by making the number of votes consistently two digits long so I could use modulus to determine which code I would be checking for that index. I also had problems with empty strings because unassigned variables can’t be negative so str.length()-1 wouldn’t work when the length of the string was 0 but I figured it out by looking at the FAQ and fixed the problem.

Pseudocode:

Is valid uppercase state code function (takes state code string)

List of valid state codes

Checks if given state code is present in list (returns true if there is a match and false if not)

Has right syntax function (takes poll data string)

Iterates through string and if a character in the string is a number that has a non digit character before and after it, adds a 0 character before it

If poll data string length is not a multiple of five or is zero

Return false

Iterates through a string

If the the party code is not in alphabet

Return false

If the number code is not made of numbers

Return false

If the state code is not a real state code (calls is valid uppercase… function)

Return false

Otherwise return true

Compute votes function (takes poll data string, party character, and vote count integer reference)

Iterates through string and if a character in the string is a number that has a non digit character before and after it, adds a 0 character before it

If polldata has incorrect syntax

Return 1

Iterates through poll data

If one of the number codes is ‘00’

Return 2

If the party code given is not in the alphabet

Return 3

Sets the current party to the first party code in poll data string in uppercase

Iterates through poll data string

If index is a multiple of five

Set current party to the character at index in uppercase

If index corresponds to number code and current party code matches given party character

Add number code to vote count

Return 0 after iteration through string ends

Test cases:

assert(hasRightSyntax("R40TXD54CAr6Msd28nYL06UT"));//checks success

assert(!hasRightSyntax(""));//checks empty string failure

assert(!hasRightSyntax("R40TXD5CaAr6Msd28nYL06UT"));//checks poll data length failure

This will fail because zeros will be added before the 5 and the 6, making the length 26 which is not a multiple of 25 - the true error is the extra a in CaAr but is caught by string length failure

assert(!hasRightSyntax("4R0MXD54CAr6Msd28nYL06UT"));//checks state code failure

assert(!hasRightSyntax("R40TX154CAr6Msd28nYL06UT"));//checks party code failure

assert(!hasRightSyntax("eawiuasdnkfjaweni232423wjkafnkwj")); //random string check failure

votes = 25;

assert(computeVotes("R40TXD54CAr6Msd28nYL06UT", 'd', votes) == 0 && votes == 82); // successful case and ensures vote code gets set to zero before calculations

votes = -999;

assert(computeVotes("R40TXD54CAr6Msd28nYL06UT", 'Q', votes) == 0 && votes == 0); // party code never matches

votes = -999;// so we can detect whether computeVotes sets votes

assert(computeVotes("R40TX3242", '%', votes) == 1 && votes == -999); // incorrect syntax

votes = -999;// so we can detect whether computeVotes sets votes

assert(computeVotes("R40TXD54CAr0Msd28nYL06UT", 'D', votes) == 2 && votes == -999); // 0 vote code failure

votes = -999;// so we can detect whether computeVotes sets votes

assert(computeVotes("R40TXD54CA", '%', votes) == 3 && votes == -999); // non alphabet party code failure