1. Brief description of notable obstacles

In development of this project, the first obstacle is that how can I implement the functions that the spec requires. Because both *isSyntacticallyCorrect* and *tallyVotes* has multiple features under different conditions, I decide to break both functions into many sub-functions to better structure my program. And I need to clarify the different cases I might face and ensure each sub-function can handle complex cases. Then, in development of my sub-functions, the most common problem I faced is “out of boundaries”, which the index I used in *string.at(index)* is larger than the length of the string. And the debugging process takes a long time since I need to test each sub-function separately with different cases, and I then integrate them into the required *isSyntacticallyCorrect* and *tallyVotes* functions*.* Finally, I used the function *assert* in the main to test my program.

1. A description of the design of the program

bool isSyntacticallyCorrect(string pollData):

store the *pollData* to local variable *data*

call function *data.empty()*

if the string is empty, return true

call function *hasOnlyLetterAndNumber(string data)*

if the string contains characters other than number and letters, return false

call the function *toUpper(string data)* to convert the whole string to upper case

call the function *isValidStateForecast(string data)*

if any state forecase is not valid, return false

if all if statements are not executed, return true

int tallyVotes(string pollData, char party, int& voteTally):

store the *pollData* to local variable *data*

call the function *toUpper(string data)* to convert the whole string to upper case

call function *isSyntacticallyCorrect(string data)*

if the poll data string is not syntactically correct, return 1

call function *isalpha(string party)*

if the party code is not a letter, return 2

call the function *notZeroForecastPredicts(string data)*

if one state forecast predicts zero electoral votes for that state, return 3

check the whether the poll data string is empty

if the string is empty, set *voteTally* to 0 and return 0

call the function *countVote(string data, char party)*

pass the return value of this function to *voteTally*

return 0

bool isValidUppercaseStateCode(string stateCode):

check whether the state code is valid upper case state code

string toUpper(string pollData):

store the *pollData* to local variable *data*

for every character in the string:

convert the character to upper case

return the converted string

bool isValidStateForecast(string pollData)

store the *pollData* to local variable *data*

while the index of the character is less than the size of the string:

set string *stateCode* to empty

set string *electoralVotes* to empty

set int *lessThanTwo* to 1

set string *partyCode* to empty

takes the characters in the string as long as the current character is a digit

append the digits to *electoralVotes*

if *electoralVotes* has a size of 0 or a size more than 2:

return false

takes the next 2 characters in the string and append them to *stateCode*

if *stateCode* is not valid upper-case state code:

return false

takes the characters in the string as long as the current character is a letter

append the characters to *partyCode*

if the *partyCode* is not valid:

return false

if all if statements are not executed:

return true

bool notZeroForecastPredicts(string pollData):

while the index of the character is less than the size of the string:

set string *predicts* to empty

pass the current characters if it is an letter

takes the characters in the string as long as the current character is a digit

append these digits to *predicts*

if the string *predicts* is “00” or “0”:

return false

if all if statements are not executed:

return true

int countVote(string pollData, char party)

set int *totalVotes* to 0

convert the party code to upper case

while the index of the character is less than the size of the string:

set string *votes* to empty

takes the characters in the string as long as the current character is a digit

append these digits to *votes*

increase the index by 2 to pass the state code

if the following character is the party code indicated in the parameter:

convert *votes* to int by calling the function *stringToInt(string str)*

add the number to *totalVotes*

increase the index by 1 to pass the party code

return *totalVotes*

int stringToInt(string str):

set *value* to 0

if the size of *str* is 2:

takes the first digit and multiply it by 10 and add it to the second digit

pass the value to *value*

if the size of *str* is 1:

pass the value to the digit to *value*

return value

1. A list of the test data

Correct poll data srting

isSyntacticallyCorrect("38TXR55CAD");

Incorrect state code

isSyntacticallyCorrect("38MXR55CAD");

Poll data srting in incorrect order

isSyntacticallyCorrect("TXR55CAD");

Incorrect party code

isSyntacticallyCorrect("38TXA55CAD");

Additional number after a state forecast

isSyntacticallyCorrect("38TXR55CAD2");

No number in a state forecast

isSyntacticallyCorrect("38TXCAD");

a number between state code and party code

isSyntacticallyCorrect("1TX2R55CAdD");

poll data srting with upper case and lower-case number

isSyntacticallyCorrect("38TXr55caD");

space or other character in the poll data srting

isSyntacticallyCorrect("38 MXr55caD");

super long poll data srting

isSyntacticallyCorrect("38TXR55CAD6Msr29nYd06UTL");

correct input

votes = -999;

tallyVotes("38TXR55CAD6Msr29nYd06UTL", 'd', votes);

invalid party code

votes = -999;

tallyVotes("38TXR55CAD", '%', votes);

invalid poll data string

votes = -999;

tallyVotes("38TXR55CRD", '%', votes);

empty poll data string with invalid party code

votes = -999;

tallyVotes("", '%', votes);

empty poll data string only

votes = -999;

tallyVotes("", 'r', votes);

zero forecast

votes = -999;

tallyVotes("00TXR55CAD ", 'r', votes);

zero forecast

votes = -999;

tallyVotes("0TXR55CAD ", 'r', votes);