1)

**Notable Obstacles**

One obstacle I struggled with was the conversion of chars to ints. Eventually I solved this by hardcoding a minus 48 to convert the chars representing ints, to the integers themselves. I also had a difficult time with checking the validity of states within the middle of the code. I solved this by creating a separate function which passed by reference the value of the position within the string so I could thus check the validity while simultaneously changing the position within the string.

2)

**Pseudocode**

Function to check valid states code

Return true or false

Function to check state validity after comma

Add 2 to position

Return whether or not it is valid state code

Function to check a valid poll string

If first two words aren't valid state code

Return false

Cycle through string char by char

If the string exceeds two numbers in a row without separation of a char

Return false

If the string has two chars in a row that isn’t the state code

Return False

If the char is a comma and does not pass state code after comma function

Return false

If it is not a letter, digit or comma

Return false

If the last character in the string is a number

Return false

Otherwise return true

Function to add two chars that represent nums

Return the first char converted to an int \* 10, plus the second char converted to an int

Function to count string

If it is a valid poll string

Cycle through string char by char

If the char is a digit or two digits in a row

Add it to a variable tempCount

If the char is a letter

If the char is equal to either the upper or lowercase char passed in

Add tempCount to the seatCount

Set tempCount back to zero

If the Char is not equal to the char passed in to the string

Set tempCount back to zero

If it is not a valid poll string and not a letter passed in

Return 1

If it is not a letter that was passed in

Return 2

If it is not a valid poll string that passed in

Return 1

Otherwise return 0

3)

**Testing**

**For ValidPollString Function:**

Cerr << isValidPollString(“NY”) << endl;

Cerr << isValidPollString(“nY”) << endl;

Cerr << isValidPollString(“Ny”) << endl;

Cerr << isValidPollString(“ny”) << endl;

To check if valid state code works in all scenarios

Cerr << isValidPollString(“Ny12d”) << endl;

Cerr << isValidPollString(“Ny12D”) << endl;

To check if ending on letter is valid

Cerr << isValidPollString(“Ny12d15r”) << endl;

Cerr << isValidPollString(“Ny12D6g”) << endl;

To check that multiple party results can be put in the string

Cerr << isValidPollString(“Ny12d,Wa,Vt10d”) << endl;

Cerr << isValidPollString(“Ny12D,vT20r”) << endl;

To check that multiple states can be put in and it still returns true

Cerr << isValidPollString(“Ny12d&”) << endl;

Cerr << isValidPollString(“Ny12D, Vt17”) << endl;

To check that it is false if the string includes a char that is not a letter/not a digit/not a comma

Cerr << isValidPollString(“Ny123d”) << endl;

Cerr << isValidPollString(“Ny12d15R,Ca101r10i”) << endl;

To check that having any more than 2 numbers in a row is false

Cerr << isValidPollString(“Ny12dd”) << endl;

Cerr << isValidPollString(“Ny12d15R,Ca18rr10i”) << endl;

To check that having a party without a number beforehand is invalid

Cerr << isValidPollString(“mM12d”) << endl;

Cerr << isValidPollString(“Ny12d15R,AA18r10i”) << endl;

To check that invalid state codes return false

Cerr << isValidPollString(“Ny12”) << endl;

Cerr << isValidPollString(“Ny12d15”) << endl;

Cerr << isValidPollString(“Ny12d7R,”) << endl;

To check if ending on anything but a letter returns false

**For countSeats Function:**

Cerr << countSeats(“Ny15d”, ‘d’, seatCount) << endl;

Cerr << countSeats(“Ny15d7R”, ‘d’, seatCount) << endl;

Cerr << seatcount << endl;

To check if a valid string outputs the correct seatCount

Cerr << countSeats(“Ny15d”, ‘D’, seatCount) << endl;

Cerr << countSeats(“Ny15d7R”, ‘r’, seatCount) << endl;

Cerr << seatcount << endl;

To check if a valid string outputs the correct seatCount with different upper and lowercases

Cerr << countSeats(“Ny15d,Vt6g0p”, ‘G’, seatCount) << endl;

Cerr << countSeats(“Ny15d7R,CA8r”, ‘R’, seatCount) << endl;

Cerr << seatcount << endl;

To check if a valid string with multiple states outputs correct seatcount

Cerr << countSeats(“Ny15d”, ‘x’, seatCount) << endl;

Cerr << countSeats(“Ny15d7R”, ‘p’, seatCount) << endl;

Cerr << seatcount << endl;

To check that if the seatCount is unchanged, it outputs as zero

Cerr << countSeats(“Ny15d,AA”, ‘r’, seatCount) << endl;

Cerr << countSeats(“Ny15d”, ‘&’, seatCount) << endl;

To check that if one of the string/char is invalid, seatCount remains unchanged

Cerr << countSeats(“Ny15d”, ‘ ’, seatCount) << endl;

Cerr << countSeats(“Ny15d”, ‘&’, seatCount) << endl;

To check that is the char inputted isn’t a letter it returns 2

Cerr << countSeats(“Ny15”, ‘ ’, seatCount) << endl;

Cerr << countSeats(“Ny15d,”, ‘&’, seatCount) << endl;

To check that if the string and char are invalid, the function returns 1

Cerr << countSeats(“Ny15”, ‘d’, seatCount) << endl;

Cerr << countSeats(“Ny15d,”, ‘f’, seatCount) << endl;

To check that if the string is invalid, it returns 1

Cerr << countSeats(“Ny15t10h”, ‘d’, seatCount) << endl;

Cerr << countSeats(“Ny15d,Vt7f”, ‘F’, seatCount) << endl;

To check that if they are both valid the function returns 0