### Notable Obstacles

1. Starting. When I first looked at the spec, I thought there were so many cases to cover that I didn't even want to start. I did not start right away because I was afraid to tackle this project, but through Professor Smallberg’s advice I started to incrementally develop it and it went okay.
2. Accessing information out of bounds. While counting seats, I decrement my position variable in order to check the digit or digits before the alphabet character that is identified as a party. What I did not realize is that if there was a letter in the first position(including all valid poll data strings) then I would be asking my function to check (position - 1) which was out of range. This resulted in a script error. I fixed this by putting my for loop within an if statement that only decrements if the position is greater than 0.

### Pseudocode

finds start of poll string

if invalid state code

return false

otherwise, increment position twice

while position is less than string length

check if 3 integers in a row

check for comma

check for non-digit or non-alphabetic character

check for 3 letters in a row

if digit or letter is found increment position

otherwise, return false

if conditions pass is valid poll string

to count seats

if not a valid poll string return 1

if a letter is not passed into party return 2

repeatedly:

while position isn’t at start of string

if letter matches party parameter and a digit is before the letter

decrement while still a digit

add one to vote count length

convert string to integer

add number of votes to variable

set seat count to number of votes

return 0

### Test Cases - Why I tested these cases

assert(isValidPollString("CT5D,NY9R16D1I,VT,ne3r00D"));check given case

assert(isValidPollString("CT5D"));one state forecast with party results

assert(isValidPollString("CA,ca"));two state forecasts with no party results

assert(isValidPollString("MI"));one state forecast

assert(isValidPollString("NY10R22D1I,VT1R,NJ19D5R4D,Mi6R"));long poll string

assert(isValidPollString(" "));empty string

assert(!isValidPollString("KS4R, NV3D1R"));space character

assert(!isValidPollString("CT5+D,NY9R16D1I,VT,ne3r00D"));+ character

assert(!isValidPollString("cT555D,NY9R16D1I,VT,ne3r00D"));three integers

assert(!isValidPollString("cT5D,NY999R1D1I,VT,ne3r00D"));three integers (not first forecast)

assert(!isValidPollString("cT5D,NY9R6D1I,VT,ne3r100D"));3 integers (last forecast)

assert(!isValidPollString("ZT5D,NY9R16D1I,VT,ne3r00D"));invalid state code

assert(!isValidPollString("KS 4R"));space character

assert(!isValidPollString("2d"));invalid state code

assert(!isValidPollString(",cT52D,NY19R6D13I,VT,ne32r10078888D"));comma first

assert(!isValidPollString("cT5D,NYc9R6D1I,VT,ne3r100D"));3 letters in a row

seats = -999;test to see if counts correctly on valid string

assert(countSeats("CT5D,NY9R9D1I,VT,ne3r00D", 'd', seats) == 0 && seats == 14);

seats = -999; see seats of party not in string

assert(countSeats("CT5D,NY9R16D1I,VT,ne3r00D", 'k', seats) == 0 && seats == 0);

seats = -999;not letter character in party parameter

assert(countSeats("CT5D,NY9R16D1I,VT,ne3r00D", '+', seats) == 2 && seats == -999);

seats = -999;check to see if it calcs correctly again

assert(countSeats("CT05D,NY9R16D1I,VT,ne3r10D", 'd', seats) == 0 && seats == 31);

seats = -999; “” \*”” is “same reason as above”

assert(countSeats("CT05D,NY9R10D1I,VT,ne3r20D", 'd', seats) == 0 && seats == 35);

seats = -999;””

assert(countSeats("OR01D", 'd', seats) == 0 && seats == 1);

seats = -999;with one forecast

assert(countSeats("OR10D", 'd', seats) == 0 && seats == 10);

seats = -999;party letter is first letter of state forecast

assert(countSeats("CA", 'c', seats) == 0 && seats == 0);

seats = 0; not valid pollstring

assert(countSeats("CT511D,NY9R16D1I,VT,ne3r00D", 'd', seats) == 1 && seats == 0);

seats = -999;not valid pollstring, checks if seats really stays what was given before fail

assert(countSeats("CT5D,NY9R176D1I,VT,ne3r00D", '+', seats) == 1 && seats == -999);

seats = -100;””

assert(countSeats("CT5D,NY9R76D1I,VT,ne223r00D", ',', seats) == 1 && seats == -100);

seats = 3;””

assert(countSeats("/CT5D,NY9R76D1I,VT,ne27r10D", '-', seats) == 1 && seats == 3);

seats = -999;not valid pollstring

assert(countSeats("w", 'W', seats) == 1 && seats == -999);

seats = -999;tests empty string

assert(countSeats(" ", 'd', seats) == 0 && seats == 0);

seats = -999;tests empty string with a invalid party parameter as well

assert(countSeats(" ", '+', seats) == 2 && seats == -999);

Handled all cases as intended.