1. Obstacles:
   1. I first wrote the function isValidPollString to only be able to check one state forecast so it took me some time to figure out how to to loop this code and make it so it keeps repeating every time a comma is entered and the next state forecast begins.
   2. When I tried testing the isValidPollString function on the g31 linux server it prompted me with errors that were happening because for example, when there were only 2 characters left, my code tried to access the third one and subtract it from pollData which prompted an error. So I added an if statement which checked the number of characters in pollData before subtracting them.
   3. In the countSeats function I forgot to set the seatcount to 0 before I started adding to it which created a logic error and caused my code to fail some test cases. I fixed this by simply setting seatcount to 0 before the addition began.
   4. The countSeats function also didn’t account for the case of the party differing from the case in the pollData, for example, countSeats(“cA8R”, ’r’, 0) didn't add 8 to the seatCount because R and r did not match. I fixed this by creating a new string and char for the pollData and party that were completely uppercase, for example, upperParty.
2. Pseudocode

isValidUppercaseStateCode

list of valid uppercase state codes separated by a '.'

check **if** the stateCode is 2 **char** **and** that it does **not** include a '.'

**and** that the 2 characters can be found in the string of codes

returns **true** **if** yes **and** **false** **if** no

isValidLowercaseStateCode

list of valid lowercase state codes separated by a '.'

check **if** the stateCode is 2 **char** **and** that it does **not** include a '.'

**and** that the 2 characters can be found in the string of codes

returns **true** **if** yes **and** **false** **if** no

isValidOtherStateCode

list of valid **case** state codes with different cases separated by a '.'

check **if** the stateCode is 2 **char** **and** that it does **not** include a '.'

**and** that the 2 characters can be found in the string of codes

returns **true** **if** yes **and** **false** **if** no

isValidPollString

**for** the number of characters in pollData grater than 0

**if** the string has less than 2 characters

**return** **false**

**for** every second character in the string

**if** its a valid stateCode

**for** every character in the string

**if** the character is a ','

**break**

**if** the next character is a digit

**if** the next character is a digit

**if** the next character is a alphabet

**else** **return** **false**

**else** **return** **false**

set stateCode to an empty string

**if** pollData has more characters left to check

remove all the characters already checked from pollData **and** one more

**else** remove only all the characters already checked from pollData

**else** **return** **false** because its an invalid state code

**return** **true** **if** everything pases

countSeats

**if** a valid pollString has not been given

**return** 1

**if** the party isn't an alphabet

**return** 2

**for** all the characters in pollData

make them all uppercase **and** store it in upperPollData

make party uppercase **and** store it in upper Party

set seatCount to 0

**for** the number of characters in upperPollData greater than 0

**for** all the characters in upperPollData

**if** the character is ','

**break**

**if** character is the same as the party

**if** the character 2 characters before the party is a digit

make the 2 digits before party into chars

convert **these** chars to an integer **and** add it to seatCount

**else**

make the character before the party to a new char

convert **this** **new** char to an integer **and** add it to seatCount

**if** the number of characters in upperPollData is more than those already checked

remove all the characters checked **and** one more

**else** remove only all the characters already checked

**return** 0

1. Test Cases:
   1. assert(isValidPollString("CT5D,NY9R16D1I,VT,ne3r00D"));
   2. assert(!isValidPollString("ZT5D,NY9R16D1I,VT,ne3r00D")); //invalid state code
   3. assert(isValidPollString("cA0t")); //just one state code with 0 as the result
   4. assert(isValidPollString("")); //empty string
   5. assert(!isValidPollString("c")); //only one character
   6. assert(!isValidPollString("wD,")); //only invalid state code
   7. assert(!isValidPollString("00000car4")); //starts with digits
   8. assert(!isValidPollString("Ny&&\*\*()#)#)\_")); //valid code but invalid forecast
   9. assert(!isValidPollString(",")); //just a comma
   10. assert(isValidPollString("mO9r5r3r,nY7v")); //multiple results from the same party
   11. assert(!isValidPollString("Mo456j")); //3 digit long result
   12. assert(!isValidPollString(",Ca")); //starts with a comma
   13. assert(!isValidPollString("Ca,,nY99y,nj07f5r4e8c")); //double comma
   14. assert(!isValidPollString(" ")); //just a space
   15. assert(!isValidPollString("mO9r9r 5r60t3r,nY7v")); //space between poll results
   16. assert(!isValidPollString("Ny77f9h7g5f4w2q1k,vt9hg")); // 2 char long party name
   17. assert(isValidPollString("mO9r9r5r60t3r,nY7v,")); //comma at the end
   18. assert(!isValidPollString("&")); //invalid state code

seats = -999; // so we can detect whether countSeats sets seats

assert(countSeats("CT5D,NY9R16D1I,VT,ne3r00D", 'd', seats) == 0 && seats == 21);

seats = -999; // to detect if countSeats changes seats and if it returns 2 for invalid party

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

seats = 3993; //to check that returns 1 for invalid pollData

assert(countSeats("cA5r33e,nY00d432f", 'r', seats) == 1 && seats == 3993);

//to check for multiple party results for the same state

assert(countSeats("CA44r3d0g9r5n8f88h5r1r", 'r', seats) == 0 && seats == 59);

//to check for multiple party results and multiple states

assert(countSeats("nJ,ar20m,AZ6n6m4m,oh", 'm', seats) == 0 && seats == 30);

seats = 60; //to check for empty string

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

//to check when pollData does not include party

assert(countSeats("NY9R16D1I", 'm', seats) == 0 && seats == 0);

//to check if a lowercase party works with an uppercase party result and for party result being double digit but starting with 0

assert(countSeats("AZ44R3D3d3D,nY01R5n01d6T", 'd', seats) == 0 && seats == 10);