**CS31 Project3 Report**

**Xuanhe Zhang (706078980)**

**Question 1: A brief description of notable obstacles you overcame.**

1.When I implemented the *isValidPollSection()* method, I only calculated the length of the digit string and the length of the letter string to determine if each section was valid. However, this fails in the "CA10" test case, where the party code is missing, but the digit parts’ length not exceed

2. To correct the problem, I added two parameters *digitPartCount* and *letterPartCount* to record the number of occurrences of digit part and letter part. Only if the two are equal will true be returned.

3.When I try to use reference, its concept confuses me, so I searched about it online and understanded the concept of it.

4.I didn’t split the poll string by “,”, so the answer is wrong. To address the problem, I wrote some helper methods to calculate each section’s seats.

5.I forgot to process the situation of empty string. When an empty string input, it will return false. However, it should return true.

**Question 2: A description of the design of your program. You should use pseudocode in this description where it clarifies the presentation.**

int countSeats(string pollData, char party, int& seatCount) {

if it is an empty string return

transform party code and string to upper letter

determine if the string and party number are valid

splitting string to sections to count each section’s seats

}

int countSection(string pollSection, char party, int& seatCount) {

find the index of party code

get the numbers of seats before it

}

bool isValidPollString(string pollData) {

if it is an empty string return

transform string to upper letter

splitting string to sections to check the validity of each section

}

bool isValidPollSection(string pollSection) {

if first two characters is not state code, return false

determine if a string is valid by recording the length of consecutive digits or letters and the number of digit parts and letter parts

if the number of digit parts and letter parts are not equal, return false.

otherwise, return true

}

bool isValidStateCode(string stateCode) {

return true if the state code is in the given range

}

bool isValidPartyCode(char partyCode) {

return true if the party code is a letter

}

**Question 3: A list of the test data that could be used to thoroughly test your program, along with the reason for each test. You don't have to include the results of the tests, but you must note which test cases your program does not handle correctly.**

assert(isValidPollString("CT5D,NY9R16D1I,VT,ne3r00D"));

//valid string

assert(isValidPollString(allStates));

//test all state poll string

assert(isValidPollString(""));

// test empty string

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

// wrong state code

assert(!isValidPollString("GOGOBRUINS!"));

//invalid string

assert(!isValidPollString("CT5D,,NY9R16D1I,VT,ne3r00")); //repetitive comma

assert(!isValidPollString("CT5DCK"));

// lack of numbers before the party code

assert(!isValidPollString(",CA10E"));

// start with comma

assert(!isValidPollString("CA10E10A10 d"));

// test special character

assert(!isValidPollString(" "));

// test special characters

assert(!isValidPollString("CA10"));

// EDGE CASE - lacking of last party code

assert(!isValidPollString("Ca10aa"));

// EDGE CASE - last party code repeat

assert(!isValidPollString("Ca10e "));

// EDGE CASE – last special character

seats = -999;

// so we can detect whether countSeats sets seats

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

seats = -999;

// so we can detect whether countSeats changes seats

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

seats = -999;

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

//test empty string

seats = -999;

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

//test empty string with wrong party code

seats = -999;

assert(countSeats("CT5D,JNY9R16D1I,VT,ne3r00D", 'd', seats) == 1 && seats == -999);

//test invalid poll string

seats = -999;

assert(countSeats("CT5D,,NY9R16D1I,VT,ne3r00D", 'd', seats) == 1 && seats == -999);

//test dobule comma

seats = -999;

assert(countSeats("#T5D,NY9R16D1I,VT,ne3r00D", 'd', seats) == 1 && seats == -999);

//test wrong state code

seats = -999;

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

//test invalid party code

seats = -999;

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

//test party code is not exist

seats = -999;

assert(countSeats("CT5D,No23d", 'a', seats) == 1 && seats == -999); //invalid state code

seats = -999;

// returns 1 if the poll string is invalid

assert(countSeats("CS31 is Great", 'a', seats) == 1 && seats == -999);

//invalid poll string

seats = -999;

assert(countSeats("CT5A,NY9R16D1I,VT,ne3r00D", 'A', seats) == 0 && seats == 5);

//test upper case party code

seats = 999;

assert(countSeats("CT5A,NY9R16D1I,VT,ne3r99a", 'A', seats) == 0 && seats == 104);

//test sum beyound 100 and start seats with 999

seats = -999;

assert(countSeats("Ct99A,NY9R16D1I,VT,ne3r99a", 'A', seats) == 0 && seats == 198);

//test different letter cases

seats = -123;

assert(countSeats(allStates, 'A', seats) == 0 && seats == 11); //test all states

seats = 500; //test all states with positive start seats

assert(countSeats(allStates, 'k', seats) == 0 && seats == 18); //test all states

seats = -999;

assert(countSeats("CA10", ' ', seats) == 1 && seats == -999);

// EDGE CASE - lacking of last party code

assert(countSeats("Ca10aa", ' ', seats) == 1 && seats == -999); // EDGE CASE - last party code repeat

assert(countSeats("Ca10e ", 'A', seats) == 1 && seats == -999); // EDGE CASE – last special character