CS31 Project 3 Report – Joyce Chen

**Notable obstacles**:

Some obstacles I encountered included converting the number characters into integers, as well as avoiding having the index go out of bounds. For the former, I realized that I could just subtract the character ‘0’ from the number character to obtain the integer that it represented. In the PDF that Smallberg provided on the website, I saw that the character ‘0’ had a code number of *x*, while a character like ‘9’ had a code number of *x + 9*. Therefore, if I subtracted ‘9’ by ‘0’ I would get the code number of 9; now, I could store this into an integer and it would give me the actual value of 9. I used this method to add up the seat counts, since the votes were represented in character format inside the poll data string.

I also spent a lot of time checking the boundaries for all my indices, since I used string methods such as s.at(index i) and had checks to see whether the “next 2” or “next 3” characters matched the requirements. It was extremely easy to miss a line of code that had the possibility of having the index going out of bounds. Therefore, I added checks such as “i + 1 < pollData.size()” into my code to prevent the compiler from whining – even though these exception errors would only happen if the poll string weren’t valid, it’s still super important because *what if the poll string weren’t valid?* Part of programming is anticipating these user errors and having conditions to deal with them.

**Description of design**:

For the isValidPollString method, I simplified the problem into these conditions:

1. An empty string would return true
2. Any character in the string must be either a digit, letter, or comma
3. If the character is a comma, it must be followed by a valid state code
4. After the state code, each of the next 2-3 characters must be a valid party result

I first checked whether the inputted string was empty, and if not, whether it had a valid 2-letter state code for the first 2 characters. Then, after those initial checks, I had a for-loop running the rest of the string from the 2nd index to the end of the string, incrementing by the integer “length” declared previously. This loop only checks for errors, so once the loop is finished, the method returns true. For each character in the string, I first checked whether it was a comma – if it was, I called the isValidUppercaseStatecode method to see whether the next 2 characters were a valid state code. “Length” would be assigned a value of 3, and the loop would increment by 3 to reach the character after the state code. I’d check whether the next 2-3 characters are a digit/letter (D3) or a digit/digit/letter (D03) – based on that, I’d set the length to 2 or 3 respectively. If none of the above happens (e.g. character is not a comma, valid state code, or follows the party results convention), then I’d return false. To sum up, I used one loop to get across the poll string, and “jumped” in increments to check whether certain substrings were valid.

Finally, for the countSeats method, I did some initial checks for whether the string was valid and if the party code was a letter. If these 2 conditions didn’t pass, then the function will return the original seat count number. If the conditions passed, I would set the seat count to 0. And once again, I would loop through the entire poll data string to find a character that matches the party code argument (not case-sensitive) – if there was only 1 digit before that character, then I would add that to the seat count directly. If there were 2 digits, I’d add 10 times the first digit and the second digit to the seat count and return the count.

**Pseudocode**:

countSeats function:

If poll data is not valid:

Return 1

Otherwise, if the party code is not a letter:

Return 2

Set the seat count to 0

Repeatedly across the entire valid poll data string:

If current character is equal to the party code:

If previous character exists and is a digit:

Add digit (convert to number) to the seat count

If previous previous character exists and is a digit:

Add digit multiplied by 10 to the seat count

Return 0

isValidPollString function:

If poll data is an empty string:

Return true

If poll data length is less than 1 or its first 2 characters don’t make a valid state code:

Return false

Declare “length” variable and set to 1 (track how much to increment in each iteration)

Repeatedly incrementing by “length” from 3rd index → end of poll data string:

If current character is a comma and next 2 characters exist:

If next 2 characters don’t make a valid state code:

Return false

Set length to 3

Otherwise, if current character and next character form a valid party result:

Set length to 2

Otherwise, if current character and next 2 characters form a valid party result:

Set length to 3

Otherwise, if none of the above applies:

Return false

Return true

**Test data**:



