Raj Kumar

Professor Smallberg

Computer Science 31

3 November 2020

Project 2 Report

**Obstacles:**

The largest obstacle was making sure every small detail of the spec was covered in my project code. I kept realizing that I was missing code that would account for a certain case of input. Next time, I will take notes on the spec to make sure I do not miss as much.

My second obstacle was realizing that I was incorrectly comparing the integer 0 to the character ‘0’ to see if there was a zero electoral votes state (even though I was specifically warned for this logic error in during lecture). I eventually found the mistake and will try to be more careful in the future.

My third obstacle was not putting the correct command to copy my poll.cpp file over to the Linux server, as I was forgetting the *:Documents* at the very end of the *scp* command in command line. I ended up having to watch the recorded discussion section explaining the steps of using the Linux server to remind me what I was doing wrong.

**Pseudocode:**

Check for correct syntax (isSyntacticallyCorrect)

Repeat until no more forecasts

Check if first one or two indexes are numbers

Check validity of state code (isValidStateCode)

Check if party character is a letter

Next forecast

Check if party is letter

Check if there are no zero electoral vote states (hasZeroElectoralVotesAnywhere)

Set vote tally to zero

Repeat until no more forecasts

Find electoral vote count

Check if same party

Add electoral vote count to tally

Next forecast

It is a lot less confusing to just read the code and assert statements than to explain the inputs in this case, so I have explained why I used a test case and then showed the code for it.

**Test Data: (**All test cases were handled in a way the spec requires)

Checks if this is correctly identified as correct

assert(isSyntacticallyCorrect("38TXR55CAD"));

Checks if program sees that tX is still Texas and thus still correct

assert(isSyntacticallyCorrect("38tXR55CAD"));

Checks if having correct syntax plus another extra letter is false

assert(!isSyntacticallyCorrect("38TXR55CAD1"));

Checks if MX is recognized as not being a state

assert(!isSyntacticallyCorrect("38MXR55CAD"));

So we can detect whether tallyVotes sets votes

int votes;

votes = -999;

assert(tallyVotes("38TXR55CAD6Msr29nYd06UTL", 'd', votes) == 0 && votes == 84);

So we can detect whether tallyVotes doesn’t set votes when it is not supposed to

votes = -999;

assert(tallyVotes("38TXR55CAD", '%', votes) == 2 && votes == -999);

Checks that this is considered not correct syntax

int v = 9999;

assert(tallyVotes("38TXR55CAD1", 'D', v) == 1 && v == 9999);

assert(tallyVotes("29NYD38TXR4ID", 'D', v) == 1 && v == 9999);

Checks if 2 is returned because of a non-alphabet party input

v = 1;

assert(tallyVotes("38TXR55CAD6Msr29nYd06UTL", '@', v) == 2 && v == 1);

assert(tallyVotes("38TXR55CAD6Msr29nYd06UTL", '1', v) == 2 && v == 1);

assert(tallyVotes("38TXR55CAD6Msr29nYd06UTL", '^', v) == 2 && v == 1);

Checks if 3 is returned and votes are unchanged if there is a state with 0 electoral votes

v = -70;

assert(tallyVotes("29NDR38TXR0HID", 'D', v) == 3 && v == -70);

assert(tallyVotes("0NYD38TXR4HID", 'D', v) == 3 && v == -70);

assert(tallyVotes("29NYD0TXR4HID", 'D', v) == 3 && v == -70);

Makes sure the empty string works (returns 0) but still doesn't change votes

assert(tallyVotes("", 'D', v) == 0 && v == 0);

General Check

assert(tallyVotes("29NYD38TXR4HID", 'D', v) == 0 && v == 33);

Sees if this correctly sets v to 0 or not

v = 75;

assert(tallyVotes("29NYR", 'D', v) == 0 && v == 0);

Checks if the examples given in the spec work the same way as spec says

assert(tallyVotes("38TXR55CAD6Msr29nYd06UTL", 'D', v) == 0 && v == 84);

assert(tallyVotes("38TXR55CAD6Msr29nYd06UTL", 'R', v) == 0 && v == 44);

assert(tallyVotes("38TXR55CAD6Msr29nYd06UTL", 'L', v) == 0 && v == 6);