Rahul Reddy

CS 31

Professor Smallberg

Project 3 Report

# The hardest obstacle I overcame was when I encountered the error, “Thread 1: signal SIGABRT”, that had to do with my stoi function as well as the fact that some of loops went out of bounds. To fix this, I had to make sure that my variable that I was incrementing had bounds within the while loops, so that it would not go out of bounds. In addition, I also had trouble trying to get int tallyVotes function to work. My biggest flaw in that was that I could not get the votes to add up properly for the same party. For example, if two state forecasts were ‘R’ and the other one was ‘d’, I was not able to get the ‘R” forecasts to add up. I fixed this by changing the loops and rearranging how I wrote them. In addition, I sometimes had to check that things were not incremented more than once.

1. A description of the design of your program. You should use [pseudocode](https://web.cs.ucla.edu/classes/fall20/cs31/pseudocode.html) in this description where it clarifies the presentation.

hasOnlyNumbersAndLetters()

iterate through the string

if the element of the string is not a letter AND not a number

return false;

otherwise, return true;

isValidUppercaseStateCode()

const string codes = all the states;

return state code size as 2 AND find these specific strings

isValidStateForecast () {

iterate through the string

set variable votes equal to empty

while the string element that is being iterated through is digit

increment votes by it

increment j

if size of votes is greater than or equal to 3

is false;

if the string element that is being iterated through is a letter

is false;

set string code to empty

while less than size of pollData size and pollData element is a letter

add it to the strong

increment j;

if code size is not equal to 3)

is false;

}

set stateCode = elements 0 through 1 (total of 2 elements)

if (state code is not a valid upper case state code)

false;

}

}

return true;

}

bool isSyntacticallyCorrect(string pollData) {

if (pollData == "") {

return true;

}

if (!hasOnlyNumbersAndLetters(pollData)) {

return false;

}

if (!isValidStateForecast(pollData)) {

return false;

}

return true;

}

int tallyVotes()

//check if pollData has the correct syntax

if not correct syntax

return 1

//checks if the party character is a letter

if party is not a letter

return 2

//checks if the votes are zero for a state forecast

string votes = "";

iterate through pollData string

if (element is a digit)

increment votes by the element;

}

if (element is a letter)

while (k is less than pollData size and the element is is a letter

increment k by 1

}

Create variable voteTally and set to changing the string of the last variable to an integer

if (voteTally is 0)

return 3;

Decrement k by 1;

Set votes to empty string

// set voteTally equal to 0 and then variable equal to empty string if none of the above situations occur

Iterate through poll data

if string element is a letter

while(string element is a digit and k < size)

if (pollData.at(k) == toupper(party) || pollData.at(k) == tolower(party)) {

voteTally = voteTally + stoi(votes);

}

Increment k by 1;

Set variable equal to empty string

While k is less than pollData’s size and the element is a digit

Increment votes by the element

Increment k

}

Decrement k by 1;

}

return 0;

}

int main()

//insert test functions

}

1. 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. (This could happen if you didn't have time to write a complete solution, or if you ran out of time while still debugging a supposedly complete solution.) Notice that most of this portion of your report can be written just after reading the requirements in this specification, before you even start designing your program.

Test Cases:

1. assert(isSyntacticallyCorrect("43AZR33CAD"))

* checks if this returns false. The goal of this was to see if the function works as expected for something with 2 forecasts

1. assert(!isSyntacticallyCorrect("38MXR55CAD"))

* checks if this returns false. The goal of this was to see if the function works as expected for something with 2 forecasts

1. int votes;

votes = -999;

assert(tallyVotes("45AZR50CAD6Msr20nYd06UTL", 'd', votes) = 0 && votes = 70);

* the goal of this was to see whether or not the tallyVotes function works. This also tested to see if the votes for ‘d’ were added up properly and did not add any other party’s votes.

1. votes = -9

assert(tallyVotes("38TXR55CAD", '%', votes) = returns 2 && votes = -9);

the goal of this was to see whether or not the tallyVotes function works. This also tested to see if the votes would stay the same since the character ‘%’ was not present

1. assert(isSyntacticallyCorrect(""))

* checks if the empty string returns true (which it should)

1. votes = 389;

assert(tallyVotes("43AZZR33CAD"))

* checks to see whether this returns 1 since it is not a poll data string due to the fact that the second Z should represent the party, however there is another letter after (which should be a digit).

1. votes = 289;

assert(tallyVotes("43AZZR33CAD", ‘r’, votes))

* should return 0

1. votes = -999

assert(tallyVotes("38TXR55CAD", 'd', votes) = 55 );

* the goal of this was to see whether or not the tallyVotes function works. This also tested to see if the votes for ‘d’ were added up properly and did not add any other party’s votes

1. votes = -999

assert(tallyVotes("38TXR55CAD", '%', votes) == 2

* the goal of this was to see whether or not the tallyVotes function works. This also tested to see if the votes for ‘d’ were added up properly and did not add any other party’s votes

1. votes = -999

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

* the goal of this was to see whether or not the tallyVotes function works. This also tested to see if the votes for ‘r’ were added up properly and did not add any other party’s votes.