In order to make my program simpler, I designed an extra function to test the validity of a state forecast, which can be called in the hasProperSyntax() function

My design is like:

 Check size of string, false if <2

 check the first two char

    if not a state, return false

 check the last letter

    if not a letter, return false

 repeatedly:

    check the char str[i] of string

        if is a number, check next char(increment i by 1)

        if is an alpha

        check last digit, if is number

        increment i by 1

        else return False

An important point I realized during my design was that ‘h’!=’H’, which means, if the state code contains lower case letter, it will always be reported as “invalid”. So to make things easy, I converted all the letters in the string to upper case before testing.

However, the following doesn’t work out:

string stateForecast = "CT5D,NY9R17D1I,VT,ne3r00D";

for(char c:stateForecast)

       c=toupper(c);

cout<<stateForecast;

It should be written as:

for(int i=0;i<stateForecast.size();i++) //IMPORTANT! Convert to Uppercase

        stateForecast[i]=toupper(stateForecast[i]);

Here is my second design(the first version is in my .cpp file, written as a comment):

bool isStateForecast(string stateForecast){

    if(stateForecast.size()<2)

        return false;

    for(int i=0;i<stateForecast.size();i++) //IMPORTANT! Convert to Uppercase

        stateForecast[i]=toupper(stateForecast[i]);

    if(!isValidUppercaseStateCode(stateForecast.substr(0,2)))

        return false;

    if(!isalpha(stateForecast[stateForecast.size()-1]))

        return false;

    //use isValid as an indicator to exit the loop if stateForecast is already INVALID

    int i=2;

    bool isValid=true;

    while(i!=stateForecast.size()&&isValid){

        if(isdigit(stateForecast[i]))

        {

            i++;

            continue;

        }

        else if(isalpha(stateForecast[i])){

            if(!isdigit(stateForecast[i-1]))

                isValid = false;

            //Check whether a party result has more than two digits.

            //Not sure if this is necessary

            else if(isdigit(stateForecast[i-2])&&isdigit(stateForecast[i-3]))

                isValid = false;

        }

        else isValid=false;

        i++;

    }

    return isValid;

My test data:

1.NY //only one element

2.NY12d,CA88A, //pollData ends with a ‘,’

3.CT5D,NY9R17D1I,VT,ne3r00D //lower case state name

4.zT5D,NY9R17D1I,VT,ne3r00D //wrong state name

5.NE000D,UT101A //invalid number of seat

6.NY1d, CA1 //last character is not a party abbreviation

At first, I omitted the last two conditions above:

1. An invalid state forecast with number of seat of more than 2 digits

such as NE3r000d, which has three ‘0’

So the highlighted line:

if(isdigit(stateForecast[i-2])&&isdigit(stateForecast[i-3]))

is to test whether the number of seat exceed 2 digits

2.A state forecast that doesn’t end with a letter, such as “NY1”

So in my “assert”, I added the following test code:

assert(!hasProperSyntax("ZT5D,NY9R17D1I,VT,ne3r00D")||!hasProperSyntax("Ca001d5n,NY1"));