Project 3 Test Data

There were 106 test cases; the first 42 were worth .357 points each, the next 38 were worth .895 points each, and the last 26 were worth one point each. Each test case is represented by an assertion that must be true for you to pass that test. To run the test cases:

1. Remove the main routine from your poll.cpp file.
2. Append the following text to the end of your poll.cpp file, and build the resulting program.
3. For any test case you wish to try, run the program, providing as input the test number.

```
#include <iostream>
#include <string>
#include <cassert>
using namespace std;

bool hasProperSyntax(string pollData);
int tallySeats(string pollData, char party, int& seatTally);

void testone(int n)
{
        int s = 666;
        switch (n)
        {
                         default: {
            cout << "Bad argument" << endl;
                    } break; case  1: {
            assert(!hasProperSyntax("3"));
                    } break; case  2: {
            assert(!hasProperSyntax("#"));
                    } break; case  3: {
            assert(!hasProperSyntax("3A"));
                    } break; case  4: {
            assert(!hasProperSyntax("#A"));
                    } break; case  5: {
            assert(!hasProperSyntax("C"));
                    } break; case  6: {
            assert(!hasProperSyntax("C3"));
                    } break; case  7: {
            assert(!hasProperSyntax("C#"));
                    } break; case  8: {
            assert(!hasProperSyntax("CA3"));
                    } break; case  9: {
            assert(!hasProperSyntax("CA73"));
                    } break; case 10: {
            assert(!hasProperSyntax("CAD"));
                    } break; case 11: {
            assert(!hasProperSyntax("CA$"));
                    } break; case 12: {
            assert(!hasProperSyntax("CA7%"));
                    } break; case 13: {
            assert(!hasProperSyntax("CA73%"));
                    } break; case 14: {
            assert(!hasProperSyntax("CA738"));
                    } break; case 15: {
            assert(!hasProperSyntax("CA738D"));
                    } break; case 16: {
            assert(!hasProperSyntax("CA006D"));
                    } break; case 17: {
            assert(!hasProperSyntax("XU44D"));
                    } break; case 18: {
            assert(!hasProperSyntax("Xu44D"));
                    } break; case 19: {
            assert(!hasProperSyntax("xU44D"));
                    } break; case 20: {
            assert(!hasProperSyntax("xu44D"));
                    } break; case 21: {
            assert(!hasProperSyntax("CA 55D"));
                    } break; case 22: {
            assert(!hasProperSyntax("CA55 D"));
                    } break; case 23: {
            assert(!hasProperSyntax("AZ4DE5R"));
                    } break; case 24: {
            assert(!hasProperSyntax("AZ4D#5R"));
                    } break; case 25: {
            assert(!hasProperSyntax("AZ4D5"));
                    } break; case 26: {
            assert(!hasProperSyntax("AZ4D5#5R"));
                    } break; case 27: {
            assert(!hasProperSyntax("AZ4D5RI"));
                    } break; case 28: {
            assert(!hasProperSyntax("FL15RI"));
                    } break; case 29: {
            assert(!hasProperSyntax("FL15R@12D"));
                    } break; case 30: {
            assert(!hasProperSyntax("FL15R1"));
                    } break; case 31: {
            assert(!hasProperSyntax("FL15R1@12D"));
                    } break; case 32: {
            assert(!hasProperSyntax("FL15R12DE"));
                    } break; case 33: {
            assert(!hasProperSyntax("FL15R 12D"));
                    } break; case 34: {
            assert(!hasProperSyntax("UT4RHI2D"));
                    } break; case 35: {
            assert(!hasProperSyntax(","));
                    } break; case 36: {
            assert(!hasProperSyntax(",WY1R"));
                    } break; case 37: {
```

```
assert(!hasProperSyntax("WY1R,"));
            } break; case 38: {
assert(!hasProperSyntax("WY1R, MA8D,VT1D"));
            } break; case 39: {
assert(!hasProperSyntax("WY1R,MA8D ,VT1D"));
            } break; case 40: {
assert(!hasProperSyntax("WY1R,8D,VT1D"));
            } break; case 41: {
assert(!hasProperSyntax("WY1R,MA8,VT1D"));
            } break; case 42: {
assert(!hasProperSyntax("WY1R,GA10,VT1D"));
            } break; case 43: {
assert(hasProperSyntax(""));
            } break; case 44: {
assert(hasProperSyntax("CA"));
            } break; case 45: {
assert(hasProperSyntax("Ca4D"));
            } break; case 46: {
assert(hasProperSyntax("cA4D"));
            } break; case 47: {
assert(hasProperSyntax("ca4D"));
            } break; case 48: {
assert(hasProperSyntax("CA42D"));
            } break; case 49: {
assert(hasProperSyntax("Ca42D"));
            } break; case 50: {
assert(hasProperSyntax("cA42D"));
            } break; case 51: {
assert(hasProperSyntax("ca42D"));
            } break; case 52: {
assert(hasProperSyntax("CA9D"));
            } break; case 53: {
assert(hasProperSyntax("CA4D"));
            } break; case 54: {
assert(hasProperSyntax("CA0D"));
            } break; case 55: {
assert(hasProperSyntax("CA89D"));
            } break; case 56: {
assert(hasProperSyntax("CA09D"));
            } break; case 57: {
assert(hasProperSyntax("CA00D"));
            } break; case 58: {
assert(hasProperSyntax("CA4d"));
            } break; case 59: {
assert(hasProperSyntax("CA42d"));
            } break; case 60: {
assert(hasProperSyntax("CA4Z"));
            } break; case 61: {
assert(hasProperSyntax("CA42Z"));
            } break; case 62: {
assert(hasProperSyntax("KY5R1D"));
            } break; case 63: {
assert(hasProperSyntax("FL15R12D"));
            } break; case 64: {
assert(hasProperSyntax("GA10R4D"));
            } break; case 65: {
assert(hasProperSyntax("GA4D10R"));
            } break; case 66: {
assert(hasProperSyntax("CA11R22D3G1A7N"));
            } break; case 67: {
assert(hasProperSyntax("LA2R1D3R"));
            } break; case 68: {
assert(hasProperSyntax("WY1R,MA8D"));
            } break; case 69: {
assert(hasProperSyntax("WY1R,MA8D,ID2R"));
            } break; case 70: {
assert(hasProperSyntax("WY1R,MA,ID2R"));
            } break; case 71: {
assert(hasProperSyntax("CA55D,KS10R,TX20R"));
            } break; case 72: {
assert(hasProperSyntax("CA20D4R,KS4R,CA19D10R"));
            } break; case 73: {
assert(hasProperSyntax("AL1D7R,AZ4D5R,AK4R,CA14R39D,CO3D4R"));
            } break; case 74: {
assert(tallySeats("3#QQ## QQ####", 'D', s) == 1);
            } break; case 75: {
tallySeats("3#QQ## QQ####", 'D', s);
assert(s == 666);
            } break; case 76: {
assert(tallySeats("WV5R", '5', s) == 2);
            } break; case 77: {
tallySeats("WV5R", '5', s);
assert(s == 666);
            } break; case 78: {
assert(tallySeats("TX38R", '5', s) == 2);
            } break; case 79: {
tallySeats("TX38R", '5', s);
assert(s == 666);
            } break; case 80: {
int r = tallySeats("3#QQ## QQ####", '%', s);
assert(r == 1  ||  r == 2);
            } break; case 81: {
assert(tallySeats("SD3R", 'R', s) == 0  &&  s == 3);
            } break; case 82: {
assert(tallySeats("SD3r", 'R', s) == 0  &&  s == 3);
            } break; case 83: {
assert(tallySeats("SD3R", 'r', s) == 0  &&  s == 3);
            } break; case 84: {
assert(tallySeats("SD3r", 'r', s) == 0  &&  s == 3);
            } break; case 85: {
assert(tallySeats("NY29D", 'D', s) == 0  &&  s == 29);
```

```
                } break; case 86: {
        assert(tallySeats("NY29d", 'D', s) == 0  &&  s == 29);
                } break; case 87: {
        assert(tallySeats("NY29D", 'd', s) == 0  &&  s == 29);
                } break; case 88: {
        assert(tallySeats("NY29d", 'd', s) == 0  &&  s == 29);
                } break; case 89: {
        assert(tallySeats("UT6L", 'D', s) == 0  &&  s == 0);
                } break; case 90: {
        assert(tallySeats("WA11G", 'D', s) == 0  &&  s == 0);
                } break; case 91: {
        assert(tallySeats("WA3G1D5L2R", 'L', s) == 0  &&  s == 5);
                } break; case 92: {
        assert(tallySeats("WA03G01D05L12R", 'L', s) == 0  &&  s == 5);
                } break; case 93: {
        assert(tallySeats("WA03G01D05L12R", 'R', s) == 0  &&  s == 12);
                } break; case 94: {
        assert(tallySeats("LA2R6D3R", 'R', s) == 0  &&  s == 5);
                } break; case 95: {
        assert(tallySeats("KS,WY,VT,HI", 'G', s) == 0  &&  s == 0);
                } break; case 96: {
        assert(tallySeats("KS4R,WY3G,VT1I,HI2D", 'G', s) == 0  &&  s == 3);
                } break; case 97: {
        assert(tallySeats("KS14R,WY13G,VT11I,HI12D", 'G', s) == 0  &&  s == 13);
                } break; case 98: {
        assert(tallySeats("KS4R,WY3G,VT1I,HI2D", 'L', s) == 0  &&  s == 0);
                } break; case 99: {
        assert(tallySeats("IL11R,DE5G,MD7D", 'D', s) == 0  &&  s == 7);
                } break; case 100: {
        assert(tallySeats("KS4R,WY1R,MA9D,ID2R,HI2D", 'R', s) == 0  &&  s == 7);
                } break; case 101: {
        assert(tallySeats("AL1D6R,CT5D,KY5R1D,MI9R4D,NJ7D5R", 'R', s) == 0  &&  s == 25);
                } break; case 102: {
        assert(tallySeats("CA39D14R,FL15R11D,TX11D25R", 'R', s) == 0  &&  s == 54);
                } break; case 103: {
        assert(tallySeats("MI4D4R5R,TN1D4R1D3R", 'R', s) == 0  &&  s == 16);
                } break; case 104: {
        assert(tallySeats(
            "AL1D6R,AK1R,AZ4D5R,AR4R,CA39D14R,CO3D4R,CT5D,DE1D,FL11D16R,"
            "GA4D10R,HI2D,ID2R,IL11D7R,IN2D7R,IA1D3R,KS4R,KY1D5R,LA1D5R,"
            "ME1D1R,MD7D1R,MA9D,MI5D9R,MN5D3R,MS1D3R,MO2D6R,MT1R,NE3R,"
            "NV3D1R,NH2D,NJ7D5R,NM2D1R,NY18D9R,NC3D10R,ND1R,OH4D12R,"
            "OK5R,OR4D1R,PA1D11R,RI2D,SC1D6R,SD1R,TN2D7R,TX11D25R,UT4R,"
            "VT1D,VA4D7R,WA6D4R,WV3R,WI3D5R,WY1R", 'D', s) == 0  &&  s == 190);
                } break; case 105: {
        assert(tallySeats(
            "AL1D6R,AK1R,AZ4D5R,AR4R,CA39D14R,CO3D4R,CT5D,DE1D,FL11D16R,"
            "GA4D10R,HI2D,ID2R,IL11D7R,IN2D7R,IA1D3R,KS4R,KY1D5R,LA1D5R,"
            "ME1D1R,MD7D1R,MA9D,MI5D9R,MN5D3R,MS1D3R,MO2D6R,MT1R,NE3R,"
            "NV3D1R,NH2D,NJ7D5R,NM2D1R,NY18D9R,NC3D10R,ND1R,OH4D12R,"
            "OK5R,OR4D1R,PA1D11R,RI2D,SC1D6R,SD1R,TN2D7R,TX11D25R,UT4R,"
            "VT1D,VA4D7R,WA6D4R,WV3R,WI3D5R,WY1R", 'R', s) == 0  &&  s == 239);
                } break; case 106: {
        assert(tallySeats(
            "AL1D6R,AK1R,AZ4D5R,AR4R,CA39D14R,CO3D4R,CT5D,DE1D,FL11D16R,"
            "GA4D10R,HI2D,ID2R,IL11D7R,IN2D7R,IA1D3R,KS4R,KY1D5R,LA1D5R,"
            "ME1D1R,MD7D1R,MA9D,MI5D9R,MN5D3R,MS1D3R,MO2D6R,MT1R,NE3R,"
            "NV3D1R,NH2D,NJ7D5R,NM2D1R,NY18D9R,NC3D10R,ND1R,OH4D12R,"
            "OK5R,OR4D1R,PA1D11R,RI2D,SC1D6R,SD1R,TN2D7R,TX11D25R,UT4R,"
            "VT1D,VA4D7R,WA6D4R,WV3R,WI3D5R,WY1R", 'I', s) == 0  &&  s == 0);
                }
        }
}

int main()
{
        cout << "Enter test number: ";
        int n;
        cin >> n;
        testone(n);
        cout << "Passed!" << endl;
}
```