CMPT 225 Assignment 2

Start by downloading the assignment files. This zipfile contains a makefile, a test script and inputs/ground truths, and stubs for all of the .cpp files you need. **Do not create any additional .h and .cpp files.**

- The makefile is for use with the linux program 'make', which helps compile programs.
- The test file is test.py, which is a python file that is executable simply by typing 'test.py' at the linux shell. (or './test.py' if you don't have '.' in your PATH)
- The testing inputs are the '.in' files, and the expected outputs (called *ground truths*) are in the '.gt' files.
- *Stubs* are simply placeholders, perhaps with some implementation.

Part 1 - Words

Consider the function shown below. Roughly, the function returns true if the letters in the string s are in alphabetical order, false otherwise.

```
//  PRE: s is composed of letters from the English alphabet, with no oth


bool isInAlphabeticalOrder(string s) {
        int length = s.size();
        for (int i = 0; i < length - 1; ++i) {
                if (s[i] > s[i+1]) {
                        return false;
                }
        }
        return true;
}
```

> ***ASCII***
>
> *An exact description is that the function isInAlphabeticalOrder returns true if the characters in the string s are in ASCII order; this differs from alphabetical order in that in ASCII order, all capital letters come before all lower-case ones.*
>
> *ASCII is the character encoding we most often use for storing and manipulating text on computers. It assigns a 7-bit code to the common characters used in English writing, including numbers, basic math symbols, and punctuation. Inside memory, we commonly store the 7-bit ASCII code inside the lower bits of an 8-bit byte. Ask a search engine about "ASCII" to find a chart of which codes (numbers) are assigned to which characters.*

Your task

We will be concerned with how many times the character comparison (s[i] > s[i+1]) is executed. First, implement the isInAlphabeticalOrder function in C++ in the file *words.cpp*. The *makefile* contains a definition for *words*. You can build the executable *words* for this part of the assignment by running "make words". ("make" or "make all" will build both *words* and the executable for the second part of the assignment.)

Determine the following for the English words listed in file *wordlist* (do not convert to lowercase).

1. The average length of a word
2. The average number of character comparisons performed by *isInAlphabeticalOrder*
3. The average number of character comparisons as a function of the word length.

You will need to add code to *isInAlphabeticalOrder* (or create a new function) to help you determine these values.

Note that *words.cpp* contains code for reading *wordlist*.

- Edit the file word_answers.txt to contain your answers to 1., and 2. above.
- The code in *words.cpp* writes your answer to 3. into a file *average_comps.txt*. Use the provided script *comps.p* to plot this using gnuplot. Running *gnuplot comps.p* will produce an image file *average_comps.png* with a plot in it.

```
uname@hostname: ~$ ls average_comps.txt
average_comps.txt
uname@hostname: ~$ gnuplot comps.p
uname@hostname: ~$ ls average_comps.png
average_comps.png
```

Submit word_answers.txt and the gnuplot output average_comps.png with your C++ files from Part 2.

Part 2 - Mode

Write a C++ function that obtains the mode of a set of integers stored in an array. Recall that the mode of a set is the most frequently occurring element.

Please use the provided file mode.cpp, and fill in the function mode. Note: you must write any auxialliary functions you use, and may not include any external libraries to help (other than iostream and fstream). The *makefile* contains a definition for *mode*. You can build the executable *mode* for this part of the assignment by running "make mode".

Testing

The zipfile contains a testing script, test.py. You should run this, and other test cases, to verify correctness of your mode function.

Grading

The assignment is worth 10% and marks are allocated to the assignment as follows:

- Part 1 4%
- Part 2 4%
- Coding style (memory management, choice of algorithms, use of functions and loops, code indentation and spacing, comments, and variable naming) 2%

Submission

You should submit your assignment online to the CourSys submission server. You should submit the following:

- Modified *words.cpp*
- Modified *word_answers.txt*
- Plot of number of comparisons as a function of n *average_comps.png*
- Modified *mode.cpp*

The assignment is due at **11:59pm on Monday July 8**.