Assignment 4 Code Peer Reviews Due: Thursday, 2/21/19, 11:59pm
Assignment 4 Code Back Evaluations: Sunday, 2/24/19, 11:59pm

Go to Peerceptiv and complete your 3 peer reviews by Thursday at midnight. After Thursday at midnight, you will evaluate how helpful your peer reviews are by Sunday at midnight.

Assignment #5: 1D Array and Recursion (150 pts)
Design Due: Sunday, 2/24/19, 11:59pm
Design Peer Reviews Due: Thursday, 2/28/19, 11:59pm
Code and Design Back Evaluations: Sunday, 3/3/19, 11:59pm

Make sure you demo Assignment #4 within two weeks of the due date to receive full credit. If you go outside the two-week limit without permission, you will lose 50 points. If you fail to show up for your demo without informing anyone, then you will automatically lose 10 points.

(10 pts) Assignment 4 Reflective Post-Peer Review Survey:
http://oregonstate.qualtrics.com/jfe/form/SV_5b9JiqnokplOIuN

This assignment consists of two mini-programs, word_frequency.cpp and fractals.cpp. You will need to zip both files and upload a zipped file with both programs to Peerceptiv.

Part 1: Word Frequency

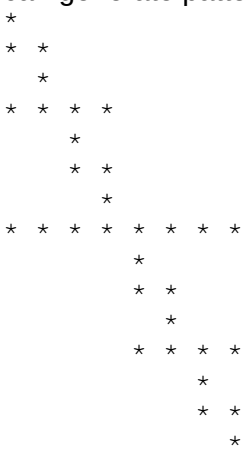
Problem Statement:
You are tasked to write a program that will count the frequency of the given words in a particular sentence or a paragraph. The program will first ask the user for a string input that is a sentence (or paragraph), then ask for N words that the user wants to search for in the sentence (or paragraph), and output the frequency for each. The sentence/paragraph must be a C-style string (array of characters ended by the null character, '\0'), but the array of words can be C++ strings.

The N words must be entered at one time before searching to see if the words are in the sentence/paragraph. In other words, you cannot ask for a word, search its frequency and then ask for another word. You must ask for all words before searching for the words. You must use a dynamic array allocated on the heap!!! You will not be given credit for a variable length array, which is not dynamically allocated on the heap, i.e. `string array[num_words]`; In addition, you must not have a memory leaks (use valgrind to help)!

*Note: Word frequency is case insensitive. E.g. "the" and "The" are the same.

(10 pts) Extra Credit:
Use an array of C-style strings for the words, instead of C++ strings. You do not have to do both. Either make the words C-style strings for extra credit OR an array of C++ strings.

Part 2: Recursive Fractals
Examine this pattern of asterisks and blanks, and write a recursive function called **pattern()** that can generate patterns such as this:



With recursive thinking, the function needs only seven or eight lines of code (including two recursive calls). Your function prototype should look like this:

```
// Description:
// The longest line of the pattern has n stars beginning in column i of the output.
// For example, the above pattern is produced by the call pattern(8, 0).
// Precondition: n is a power of 2 greater than zero.
// Postcondition: A pattern based on the above example has been printed.
void pattern(int n, int i);
```

Hint: Think about how the pattern is a fractal. Can you find two smaller versions of the pattern within the large pattern? Here is some code that may be useful within your method:

```
// A loop to print exactly i spaces:
for (k = 0; k < i; k++) cout << " ";
// A loop to print n asterisks, each one followed by a space:
for (k = 0; k < n; k++) cout << "* ";
```

(10 pts) Extra Credit:
Try to create the same fractal program iteratively!!! Submit the code for this, and it has to work for N stars and the points are all or nothing (no partial points)!

(50 pts) Design Document – Due Sunday 2/24/19, 11:59pm
Refer to the Example Polya Document - Polya_template.pdf

You need to provide a design for both parts/programs!

Step 1: Understanding the Problem/Problem Analysis. (15 pts)
Do you understand everything in the problem? List anything you do not fully understand, and make sure to ask a TA or instructor about anything you do not understand.

- What are the user inputs/requirements, program outputs, etc.? (5 pts)
- What assumptions are you are making? (5 pts)
- What are all the tasks and subtasks in this problem? (5 pts)

Step 2: Program Design. **(25 pts)**

- What does the overall big picture of this program look like? (flowchart or pseudocode) (20 pts)
    - What data do you need to create, when do you read input from the user?
    - What are the decisions that need to be made in this program?
    - What tasks are repeated?
    - How would you modularize the program, how many functions are you going to create, and what are they?
- What kind of bad input are you going to handle? (5 pts)

Based on your answers above, list the **specific steps or provide a flowchart** of what is needed to create. Be very explicit!!!

Step 4: Program Testing. **(10 pts)**

Create a test plan with the test cases (bad, good, and edge cases). What do you hope to be the expected results?

- What are the good, bad, and edge cases for ALL input in the program? Make sure to provide enough of each and for all different inputs you get from the user.

(80 pts) Program Code – Due Sunday 3/3/19, 11:59pm

(80 pts) Implementation Requirements:

- Must produce two working programs that follows each rule stated above
- Your user interface must provide clear instructions for the user and information about the data being presented
- Use of one-dimensional array/C-style strings is required
- Use of references or pointers is required
- Your program should be properly decomposed into tasks and subtasks using functions. To help you with this, use the following:
    - Make each function do one thing and one thing only
    - No more than 15-20 lines inside the curly braces of any function, including main(). Whitespace, variable declarations, single curly braces, vertical spacing, comments, and function headers do not count.
    - Functions over 20 lines need justification in comments
    - Do not put multiple statements into one line
- No global variables allowed (those declared outside of many or any other function), global constants are allowed
- goto is NOT allowed
- You must not have any memory leaks!
- Make sure you follow the style guidelines, have a program and function headers with appropriate comments, and be consistent

(10 pts) Extra Credit:

Keep a journal of errors you run into using pointers in this assignment. You must record 1) whether the error was at compile time or runtime, 2) what the error was, 3) the line number it occurred on, 4) what you did to fix it, and 5) why this fixed it.

(10 pts) Assignment 5 Predictive Pre-Peer Review Survey:

http://oregonstate.qualtrics.com/jfe/form/SV_aeH9GJKWISJJu2F

Electronically submit your Design Document by the design due date and your C++ programs (zip two .cpp files together) by the code due date using Peerceptiv.