Assignment #4 – Strings, Functions, and Recursion
Due: Sunday, 11/08/15, 11:59pm

Grading: EVERY assignment in this course is graded by demoing your work for 10 minutes with a TA. You are required to meet with a TA within one week from the due date to demo. You are penalized for failure to see a TA within the week or missing a scheduled appointment. In either case, if you are within 1 day (24 hours) of the deadline, you lose 10 points. If you are within 7 days (1 week) of the deadline, then you lose 25 points, anything outside of a week from the deadline to demo is an automatic 50 point deduction. Your job is to convince the TA that your program works correctly, i.e. show your TA how to use/break your program☺

(50 pts) Play a Game: Hangman
You will implement the program for playing Hangman. The game must allow a user to enter the secret message (possibly containing blanks), and print the number of dashes/slots for the message (spaces do not get dashes, just the space). The game will ask the user to guess letters to determine the message, and after each guess, the program must print out 1) the number of letters found for the guess, 2) the letters that have been guessed, 3) how many incorrect guesses out of however many are allowed**, 4) the message with the correctly guessed slots and blank slots, and 5) if the user guessed all the letters in the message, then a winner message.

Requirements:
- Only use C++ strings, no need for an array! Hint: use two strings!
- Your program must ignore cases in the message.
- Each function, including main, may not have more than 10 lines of code (this doesn’t include comments and blank spaces!).
- You are not allowed to use global variables.

Read the C++ string documentation to help with this assignment:
http://www.cplusplus.com/reference/string/string/?kw=string

**You can decide how many incorrect guesses a user gets before they lose. In addition, you can use the system(“clear”); command from the <cstdlib> to clear the screen, after the user enters their message. Yes they can scroll above to see the message, but we aren’t cheaters!!!

Example Run:
Enter a message: Hi
The message is _ _
Guess a letter: t

There is no t.
You have guessed: t
You have 1 incorrect out of 5.
The message is _ _
Guess a letter: h

There is 1 h.
You have guessed: t h
You have 1 incorrect out of 5.

The message is h _
Guess a letter: i

There is 1 i.
You have guessed: t h i
You have 1 incorrect out of 5.

The message is hi
You won!!!

(50 pts) 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:

```c
// 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:

```cpp
// 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 algorithm as pseudocode or code for this!

(10 pts) **Program Style/Comments – For both programs!!!**
In your implementation, make sure that you include a program header in your program, in addition to proper indentation/spacing and other comments! Below is an example header to include. Make sure you review the style guidelines for this class, and begin trying to follow them, i.e. don’t align everything on the left or put everything on one line!

http://classes.engr.oregonstate.edu/eecs/fall2015/cs161-001/161_style_guideline.pdf

```
/****************************
** Program: hangman.cpp
** Author: Your Name
** Date: 11/05/2015
** Description:
** Input:
** Output:
***************************/
```

(10 pts) **Testing – For both programs!!!**
Report any checking/self-reflection you did while solving the problem. For instance, how did you make sense of the output from the implementation? This includes things such as using a calculator to make sure the output is correct, testing to make sure your code executes correctly and behaves the way you expect under specific circumstances, using external sources of information such as the internet to make sense of the results, etc. In addition, you will provide us a test plan!

You will be graded on how thorough your test plan is. Make sure you think about input you hope works and input that won’t work. Your program doesn’t have to handle input that doesn’t work!!!

Please see the example of a test plan: Polya_template.docx

Electronically submit your two C++ program (.cpp file, not your executable!!!) and test plan, as a pdf, by the assignment due date, using TEACH.
**NOTE:** The easiest way to upload your program from ENGR to TEACH is to map a network drive to your home directory on ENGR. Mac or Windows, See: http://engineering.oregonstate.edu/computing/fileaccess/