Assignment #4
A Cipher Only a CS Person Would Understand and Recursive Fractals
Due: Sunday, 11/05/17, 11:59pm

Grading: Every assignments in this course is graded by demoing your work for 10 minutes with a TA. You are required to **meet with a TA within two weeks of the due date** to demo. You can schedule a demo with a TA on the **home page** in the far right column of the bottom table labeled "Grading Hours".

- **Demo Outside 2 Weeks:** Assignments that are not demo'd within the acceptable time period will be subject to a 50 point deduction.
- **Demo Late Assignments:** Late assignments must still be demoed within the two week demo period beginning from the assignment's original due date.
- **Missing a Demo:** If you miss your demo with a TA, you will receive a 10 point (one letter grade) deduction to that assignment for each demo missed.

Your job is to convince the TA that your program works correctly, i.e. show your TA how to use/break your program☺

(60 pts) Problem Statement: You will write a program that asks the user for **a string of characters** and prints the encrypted message back to the user as a string of ASCII values in base 2 **OR** asks the user for **a string of ASCII values in base 2** and prints the decrypted string of characters corresponding to those values back to the user. This program will use functions for converting numbers, getting input from the user, converting input from the user, and error checking.

- Ask the user if they want to encrypt a message entered as string of text or decrypt a message given as a string of ASCII characters in base 2.
- If the user wants to encrypt a message of text, then read a line of text that can include spaces, e.g. “jen” or “jen mocello”.
    - Print the user’s encrypted message back to the user in binary.
- If the user wants to decrypt a message of characters in binary, then read a string of 1s and 0s from the user.
    - Print the decrypted message in corresponding text.
- **Continue to prompt the user for encrypting or decrypting until the user doesn’t want to do this anymore.**

- **Print an error message when:**
    - The encrypted message (binary) cannot be separated into bytes
    - The encrypted message (binary) isn’t 1s and 0s
    - Since our messages are in English, we are not using the extended ASCII chart (128-255). Make sure the binary numbers will produce characters within 0-127.
- **Recover from the error by re-prompting the user for a new binary number after their error, until they give a good binary number.**

Example:
Do you want to encrypt or decrypt a message: **encrypt**
Enter the string to encrypt: **jen**
011010100110010101101110
Do you want to play again (yes or no)? **yes**

Do you want to encrypt or decrypt a message: **decrypt**
Enter the string to encrypt: **011010100110010101101110**
jen
Do you want to play again (yes or no)? **no**

The idea is to keep each function, including main, to a **maximum of 15 lines**, and this does not include variable declarations, comments, curly braces, and blank lines. Here are some functions you might want to think about creating.

- Create a bin_to_ascii() function that takes a string of 1s and 0s as input and prints/returns the binary string as an ascii character.

- Create a ascii_to_bin() function that takes an ascii character as input and print/returns the character as a binary value.

- Create a get_user_inpu() function that gets the input from the user.

- Create a check_bin_number() function checks that the user entered a binary number **meeting the three requirements specified above**.

- In addition, create any other functions you might need to properly modularize your code to 15 lines or less in each function.

(-10 pts) **Warning:** You are not allowed to use global variables in any assignment in CS 161. There isn't any practical purpose for them in this course.

(30 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:

```
// 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 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/fall2017/cs161-001/161_style_guideline.pdf

```
/*************************************************
** Program: hangman.cpp
** Author: Your Name
** Date: 11/05/2017
** Description:
** Input:
** Output:
*************************************************/
```

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/