LAB #8 – Pointers, Arrays, and Errors

Understanding Pointers:
As a part of this lab, you have to draw a picture and explain each part of a pointer. For example, when you create some pointer, `<type> *pointer;`, what do the three pieces of a pointer access, and how would you use each?

- `&pointer`
- `pointer`
- `*pointer`

You must be able to draw a picture and explain how you would use each of the three pieces of a pointer for 2 points of this lab!!!

Understanding Errors:
Here is a functional program that compares two strings supplied as command line arguments to see if they are equal. If they are equal, it returns true, and if they are not equal, then it returns false: `string_equal.cpp`. Example run:

```
./string_equal hello hello
```

You are going to deliberately create errors, and record on a piece of paper what you notice about these errors. For example, when you remove a semicolon from line 10 in the program, then the compiler gives you an error on a few lines below, at the next statement, alarming you that you have a missing semicolon before the statement on line 12. Perform each of these errors, and record your understanding of the error and why it is the error it is. After you make the error, restore the program back to functional before making another error!!!

Syntax errors
- Remove a semicolon from line 12.
- Remove the curly brace from line 4.
- Remove the brackets from the argv arguments in the call to is_equal() on line 17.
- Remove the asterisk/splat from the argv in main on line 15.
- Remove the return type, bool, from the is_equal() function on line 4.
- Remove one of the parameters in the is_equal() function on line 4.
- Comment out line 5, and declare `i` in the for loop on line 6.

Logic errors
- Put two asterisks/splats on the parameters of line 4, and put ampersand, `&`, in front of the arguments on line 17.
- Comment out line 10, and run with `./string_equal hell hello`.
- Comment out line 16, and run with `./string_equal hello`

You must be able to explain all these error for 3 points of this lab!!!
Practice Planning and Arrays/Command Line Arguments:
On a piece of paper you are going to write out all the tasks and subtasks for the following program BEFORE you write code for the program.

Write a program that takes a sentence as a command line argument and converts the sentence input by the user into pig latin. You can assume that the sentence contains no punctuation. After you convert the string, then print the sentence in pig latin. You can get a sentence as a command line argument by using quotes, i.e. 

`./pig_latin "convert this sentence"

The rules for pig latin are as follows:
- For words that begin with consonants, move the leading consonant to the end of the word and add “ay.” Thus, “ball” becomes “allbay”; “button” becomes “uttonbay”; and so forth.
- For words that begin with vowels, add “way” to the end. Thus, “all” becomes “allway”; “one” becomes “oneway”; and so forth.

Show your list of tasks/subtasks and program to a lab TA for the other 5 points of this lab.