Functions

Using functions

Part of writing code efficiently is using pre-written functions to simplify things. You're already familiar with this from the libraries you use, like cmath. In the next three drills, you are going to be using the following three functions. Copy and paste them into a cpp file before continuing.

The integer library

```
//returns the maximum of three integers, a, b, and c
int maxOfThree(int a, int b, int c)
{
  int max = a;
  if(b > max)
    max = b;
  if(c > max)
    max = c;

  return max;
}

//given two integers that represent the two non-hypoteneuse sides of a
//right triangle, computes the square of the hypoteneuse
//NOTE: Yes, I know there is a function, "hypot", in cmath, that does
//something similar, but I want an integer and no rounding
int findHypotSquare(int a, int b)
{
  return a*a + b*b;
}

//given an integer, determines if it is a perfect square
//Note: there are more efficient ways to do this, but I am using this
//method for simplicity
bool isPerfSquare(int x)
{
  for(int i = 2; i < x/2 + 1; i++)
  {
    if(i * i == x)
      return true;
  }
  return false;
}
```

The drills

For these questions, you **must** use the functions above, and any other functions you think you need. You **cannot** duplicate the functionality of any of the functions above. Well...you *can*, strictly speaking, do that. This isn't for marks, really. But you **shouldn't**, since you're here to practice functions, right?

Question 1

Have the user continually enter integers until they enter a perfect square.

Question 2

Have the user enter two integers. These two integers will be the sides of a right angle triangle. Determine if the hypoteneuse is also an integer.

Question 3

Have the user enter three integers. Determine if the three integers form a right angle triangle.

Writing functions

Question 1

Let's revisit a very common example: the pyramid of characters. Enter a number of levels, n, and a character, c. Print out a pyramid of n levels of character c. For instance, if I put in 4, x, I should see:

```
   x
  xxx
 xxxxx
xxxxxxx
```

Now, instead of writing this in main, come up with **as many functions as you can** to write this code. For instance, obtaining user input might be a function, printing a single line might be a function etc.

Question 2

Write the following functions, and test them in your main:

1. Add two numbers
2. Have the user enter a positive integer. Verify that it is positive. If it isn't positive, have the user re-enter the integer until it is positive. Return the integer.
3. Compute the modulo of an integer **without using the modulo operator**. You may decide what you want to do with negative numbers. You may also decide if you want to compute the modulo of

floating point values, and how to handle that.

Making a string library

Functions really shine when we are writing *libraries*. One of the largest, and most useful, libraries you can access is cstring, the string library. It does wonderful things like compares two strings and determines if they are equal, adds one string to another, copies one string to another etc. For the rest of this excersize, you need to know something about character arrays and strings.

Strings are based on character arrays. In fact, the following is valid code:

```
char myString[] = "Hello, world!";
```

This array is filled with the characters "Hello, world!0". What's that 0 doing there? It's called a "Null terminator" - strings need to know when they end. So when C++ was being created, someone decided that they will append the number 0 to the end of all strings. When you encounter the zero, you know you're at the end. Here is an example, called strlen, that computes how long a string is:

```
int strlen(char inString[])
{
  int i = 0;
  while(inString[i] != 0)
  {
    i++;
  }
  return i - 1;
}
```

Three things to note: first, I can pass arrays into functions, I just need to put empty [] brackets in to tell C++ that I'm using an array. Second, I returned i-1 because the ending 0 is typically not considered part of the string. Finally, I had to search for **the number 0**, not the character '0'. The character, '0', actually translates to the number 48, strangely enough.

Comparing characters can be done using <, ==, and >. For instance, if I have:

```
char a = 'A';
char b = 'B';
cout << a < b;
```

It will be true, since A comes before B in the alphabet.

OK, armed with all of this, write the following functions. Note: some of the functions you will write will use strlen, and you may decide to write additional helper functions to move you along. You must test

your functions as you see fit.

Question 1 strcmp

strcmp, or "string compare", is a function that takes two character arrays, str1 and str2, as input and outputs an integer according to the following rules:

- If str1 == str2, meaning that they are the same length and contain exactly the same characters, then 0 is returned.
- If str1 comes before str2 alphabetically, but they are the same length, -1 is returned.
- If strlen(str1) < strlen(str2), -1 is returned regardless of the contents of the strings
- If str2 comes before str1 alphabetically, but the are the same length, +1 is returned.

Test this one thoroughly. You might be surprised at what you see!

Question 2 atoi

The characters for the digits 0 through 9 are represented by the numerical values 48 through 57. For instance:

```
char c = 55;
cout << c;
```

You should see the output '6'.

The function atoi takes in a character array and converts it into an integer. Typically, the behaviour of this function is **undefined** when the input is invalid, and it's entirely up to you what to do.

Question 3 atod

Repeat the function atoi above, but instead return a double. Allow the input to contain exactly one decimal point, and handle it appropriately. Again, how to handle bad inputs is up to you.

Question 4 findChar

This function takes in a character array, str, and a single character, c. It searches str for the first instance of the character c. It then returns the index at which c is found. If c is not found, it returns a negative number.

(NOTE: this is actually a simplified version of a function called memchr that exists in cstring. You can look into it if you're interested, but it is far more complex)

Question 5 findFirstOf

This function takes in two character arrays, str1, str2. It searches str1 for **any** character in str2, and returns the index first instance of the first character from str2 that it finds in str1. It will return a negative number if no characters from str2 are found in str2.

For example:

```
char str1[] = "Helloe";
char str2[] = "abe";
cout << findFirstOf(str1, str2);
```

You should see the number 1, since str1 contains the letter e, which also exists in str2. The first time we see the letter e is at index 1.