CS 103 Lab – Variables, Expressions & Calling Functions

1 Introduction

In this lab you will gain experience writing code that uses arithmetic expressions and calls built-in functions.

2 What you will learn

After completing this lab you should be able:

- Use cin/cout to take input from the user and print output to the user
- Use expressions, define and convert between data types, and call functions

3 Background Information and Notes

None

4 Procedure

We provide skeleton code for two of the three exercises in this lab.

Download the following code samples by running these commands:

```
$ cd ~/cs103                (If using Dropbox, do cd ~/Dropbox/cs103 instead.)
$ mkdir lab-vars
$ cd lab-vars
$ wget ftp://bits.usc.edu/cs103/lab-vars.tar
```

Un-tar the files and list the new directory contents. You should see some .cpp files.

```
$ tar xvf lab-vars.tar
$ ls
```

You'll see the following partially completed files into your directory:

```
color_conv.cpp
compute_sin.cpp
```

4.1 Basic I/O, Expressions, Order of Operations Calling Functions,

In calculus, a Taylor series approximates arbitrary functions by polynomials. E.g., the function sin(x) can be approximated using the series expansion:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \cdots$$

Your goal is to write a program which will compute this approximation. It should take a decimal number x as input, compute the approximation given by the first four terms of the above formula, and then print it out. (This formula assumes x is in radians. Your program should do the same, so no degrees/radians conversion is necessary.)

Open `compute_sin.cpp`, which is a skeleton program for you to fill in. Write in your name. The rest of your work will be filling in the contents of the `main` function. The comments there should help you structure your work.

Again, the formula that your program should compute is

$$x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!}$$

It uses exponents (like x^3) and factorials (like 3! which means 1x2x3, i.e., 6). E.g., if the input is 1, the output should be 1 - 1/3! + 1/5! - 1/7! ≈ 0.841468, which is a pretty good approximation to sin(1) ≈ 0.841471.

- To compute powers, do not simply multiply x by itself. Also, don't use the ^ symbol, which means "xor" in C++. Instead, use the `pow()` function. It is part of the `<cmath>` library, which is why we need that extra `#include` statement at the top. The textbook explains this function, or read online at http://www.cplusplus.com/reference/cmath/pow/
- There is no built-in factorial function in C++. However, we've provided one for you, called `fact`, in the skeleton code. We won't talk about defining our own functions for a couple more lectures. But *calling* the provided function is pretty straightforward: write `fact(3)`, or any other input in place of 3.

Make sure you use parentheses where necessary, and use an appropriate datatype for your variables. Don't assume that x is a whole number, it could be a decimal number like 1.57.

You do not need to use a loop for this exercise or any other in this lab.

After you're done editing your program in gedit, remember to compile it with

```
$ compile compute_sin
```

Then run it with

```
$ ./compute_sin
```

and see what happens when you input 1.

Ask a CP or TA for help if needed. When you're ready to check in all of your programs, demo it for a CP or TA (they'll read your code and ask you to run it on additional test cases). Upload it to the website as well: follow the link listed at http://bits.usc.edu/cs103/coursework/

4.2 Casting, Expressions

In this exercise you will write a program to convert between two color schemes. "RGB," which corresponds to "additive" colors of light like a TV or computer screen,

represents a color with 3 components Red, Green, and Blue that each range from 0 to 255. "CMYK," which corresponds to "subtractive" scheme like the wavelengths absorbed by ink, represents a color with 4 components Cyan, Magenta, Yellow, and Black that are each real values between 0.0 and 1.0. Your program will convert from RGB format to CMYK using the following conversion method:

$$\text{Define } white = \max\left\{\frac{red}{255}, \frac{green}{255}, \frac{blue}{255}\right\}$$

$$cyan = \frac{\left(white - \frac{red}{255}\right)}{white}$$

$$magenta = \frac{\left(white - \frac{green}{255}\right)}{white}$$

$$yellow = \frac{\left(white - \frac{blue}{255}\right)}{white}$$

$$black = 1 - white$$

Your program should take **3 integers**, (`red`, `green`, and `blue`) as input from the keyboard and output the corresponding values for `cyan`, `magenta`, `yellow`, and `black`. Fill in the provided skeleton program `color_conv.cpp` to achieve this.

The conversion formula includes *max*, which means the maximum (largest) of its inputs. The skeleton has `#include <algorithm>` at the top of the program, which makes C++ provide you with a function called `max` to do this. However, `max` can only compare (find the max of) two numbers at a time, so you'll have to figure out how to extend this to find the max of three numbers. Here's an example of using `max`.

```
#include <iostream>
#include <algorithm>
using namespace std;

int main() {
  cout << max(45, 63) << endl; // prints 63, it is larger
  return 0;
}
```

Example: if you input `20` (red) `80` (green) `60` (blue), the correct output should be:

```
cyan: 0.75
magenta: 0
yellow: 0.25
black: 0.686275
```

Hint: a common mistake in this exercise is **integer division** (see the lecture notes). Check in your work once it's complete.

4.3 Think, Code, Test

Write a program called `digits.cpp` to do the following. Prompt the user for an integer between 0 and 999. (You don't need to write any code to check the input.) Write code to take the single decimal value and find the 1's digit, 10's digit, and 100's digit (in that order) and store them in three separate variables. If they enter a 1 or 2-digit decimal value you should produce 0 for the leading digits. Then output the result on three separate lines to the user.

Here is a sample run:

```
Enter an integer between 0 and 999: 247
1's digit is 7
10's digit is 4
100's digit is: 2
```

Hint: Think how you can use integer division and the modulus operator to help you.

Check in your work once it's complete.

5 Review & Reflection

Take a moment and reflect on the last program. Suppose we wanted to achieve the same task of finding the separate digits of any number (i.e. 3-, 4-, 5-, etc. digits) in ascending order (from 1's to 10's to …) using a ***while*** loop? What would you do in each body/iteration? What would the condition for repeating be (i.e. the while condition)?

Write a brief English description on this paper or in a text file. You can optionally re-write your program using this approach if you like. Show this to your CP/TA when you check in.