APS 105 — Computer Fundamentals

Lab #2: Programs That Calculate
Fall 2013

Important note: The course material necessary for completing this lab will be found in your lecture notes and in the Carter text up to the end of Section 2.6.

You must use the `submitaps105f` command to electronically submit your solution by 11:59pm on **Saturday, September 21, 2013**.

Objective

In this lab you will be writing some short programs using variables and arithmetic operators. Input and output must be done using `scanf` and `printf`.

In your programs, you may only use the programming techniques presented in lectures or in Chapters One or Two of the Carter textbook. You are required to document your program with comments. Your comments should at least include your full name and student number, and a very short description of what the program does.

Also, in this assignment floating output should be printed with a minimum width of 5 characters, including the decimal point and decimal numbers. For example, to print the number 4.56 this way, use the printf format specifier "%5.2f", which will output " 4.56", adding a leading space to add up to 5 characters total.

Sample output

In the sample output examples that follow, the text that would be entered by the program user is displayed using a **bold** font. The text <**enter**> stands for the user pressing the `enter` key on the keyboard. On some keyboards, the `enter` key may be labeled `return`. When you execute your programs, the user's text will not be displayed in bold and <enter> will not be shown.

Part 1 — Currency Conversion

When travelling to another country, we're often faced with the need convert currency amounts from the foreign currency to the one we're used to. Certainly there are many apps for that! Let's make a simple one, that converts from any foreign currency to Canadian dollars. Its input will consist of the conversion rate (the number of Canadian dollars equivalent to each unit of the foreign currency) and the amount of the foreign currency to be converted.

You are to write a C program that takes as input (from the keyboard) the conversion rate and amount to be converted (both double precision floating point numbers). Your program should calculate and output the amount of Canadian dollars equivalent to an input amount foreign currency. The output dollar amount should use no more than two digits on the right of the decimal point.

Here is sample output from an execution of the program:

```
Enter the conversion rate: 0.17<enter>
Enter the amount to be converted (in foreign currency): 245.00<enter>

The amount in Canadian Dollars is:  41.65
```

Note: You can assume that the user enters valid numbers.

Your C program must go in a file named `Lab2Part1.c`.

Part 2 — Conversion to Base 2

As you launch into the world of computers, you'll notice we often mention bits (short for 'binary digit') as the way computers actually represent numbers inside the machine. Binary, also known as the base 2 representation will show up all the time as we talk about hardware and software. You are to write a program that converts a base 10 integer number (assuming it is no larger than 15) into base 2.

The input to your program is an integer number, no larger than 15. The output, on separated lines, are the four digits of the base 2 representation of that number. Below we give a sample input and output sequence for the program:

```
Enter number to convert to base 2: 13<enter>
The four digits of that number are as follows:
Most significant digit: 1
Next digit: 1
Next digit: 0
Least significant digit: 1
```

For the example above, the program convert 13 in base 10 to 1101 in base 2. Note the left-most digit of any number is called its "most significant digit", and the right-most digit is called the "least significant digit" for perhaps obvious reasons.

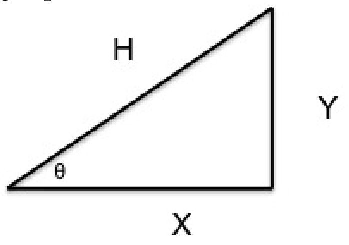
Note: You can assume that the number entered is postive, less than 16, and is an integer. Also, you should always output four bits, regardless of the size of the input number. For example, if the input number is '3', then the output four digits will be 0011.

It will be helpful to make use of the modulo operator (%) which gives the remainder after divison, as described on page 53 of the text.

Your C program must go in a file named `Lab2Part2.c`.

Part 3 — Triangle Calculations

Consider the right-angled triangle pictured below:



You are to write a program that computes the length of the hypotenuse (H) of a right-angled triangle, given the length of the other two sides, X and Y, with the Pythagorean formula:

$$H = \sqrt{X^2 + Y^2}$$

You should also compute the angle, θ, using the formula:

$$\theta = sin^{-1}(\frac{Y}{H})$$

Your program reads in the values of X and Y (which should be of type 'double' floating point numbers) and outputs the resulting H and θ. **The value of θ should be given in**

degrees (note that the asin function will return its result in radians, and so you will have to compute the conversion to degrees separately). The values of H and θ should be displayed with just 1 digit to the right of the decimal point. Here is sample output from an execution of the program:

```
    Enter X: 3.0<enter>
    Enter Y: 4.0<enter>

    The Hypotenuse Length is: 5.0
    The Angle Theta is:  53.1 degrees
```

Note: To compute $\sqrt{Z}$, use the `sqrt` function in the `math.h` library, and to compute $sin^{-1}(Z)$, use the `asin` function which is also in the `math.h` library. To use the `math.h` library, you will need to add the following line to the beginning of your program:

```
    #include <math.h>
```

Also, you need to tell `gcc` to *link* the implementation of this function to your program. You will do this simply by adding `-lm` option to the gcc command (i.e. `gcc -lm ...`). Unlike other C libraries such as `<stdio.h>` and `<stdlib.h>`, using `<math.h>` requires that you also explicitly link to its implementation by adding `-lm` to the gcc command line.

Your C program must go in a file named `Lab2Part3.c`.

Marking

This lab will be marked out of 10. Parts 1 and 3 have each 3 marks, and part 2 has 4 marks. Full marks are given if your program works correctly, fewer if not, and zero if it cannot be compiled. Late submissions or submissions with an incorrect filename will result in a mark of 0 for the entire lab. The deadline is strictly enforced, so avoid last minute submissions.

You can run a testing program, called `tester`, yourself to test the correctness of your solution. At the command line in your ECF account, run:

```
/share/copy/aps105f/lab2/tester
```

in the same directory as your solution file. The testing program will use a number of test cases to test your solution, and report `success` if your solution produces output that is **identical** to the expected output.
If your output does not match the expected output, please look carefully at the output of the tester for the expected output.

A marker program will be used to automatically mark your lab. The marker program will use **the last version** of the lab files you submitted using the `submitaps105f` command. Some of these test cases will be used by the marking program as well, but the marking program will also be using other test cases that are not included in the testing program to test the correctness of your program. This implies that even though you do not have access to the marker program before the submission deadline, you will obtain at least partial marks if all of the test cases in the testing program report `success` with your solution. After the submission deadline, you can access the marker program by running the following command at the command line in your ECF account:

```
/share/copy/aps105f/lab2/marker
```

The mark you receive for each part will be the same as reported by the marker program for the file you submit. Late submissions or submissions with the incorrect filename will result in a mark of 0 for the whole lab. The deadline will be strictly enforced, so avoid last minute submissions.

Your mark for this lab will contribute 1% to your mark in the course.

What To Submit

When you have completed the lab, use the command

```
% submitaps105f 2 Lab2Part1.c Lab2Part2.c Lab2Part3.c
```

to submit your files. You can also submit the files individually after you complete each part of the lab. Make sure you name your files exactly as stated (including lower/upper case letters). Failure to do so will result in a mark of 0 being assigned. You may check the status of your submission using the command

```
% submitaps105f -l 2
```

where `-l` is a hyphen followed by the letter *'ell'*.

You can also download a copy of your submission by running the command:

```
/share/copy/aps105f/lab2/viewsubmitted
```

Good Luck!