AE1PGA Lab 2

Below are some tasks that will give you an opportunity to practice using the features and concepts we've discussed in the last few lectures. Often these tasks have some unexpected complications, so don't skip them just because they look easy when you read them. Some tasks will require you to go off and read/search in a little more depth about a particular topic (eg, reading a bit more about how `scanf` can be used in the question below). This is normal when writing new kinds of programs and all these little pieces of extra knowledge you pick up will give you a bigger toolbox to use in later programs.

If you finish these quickly, just ask one of the lab instructors for some more things to do and we can give you some more advanced material. Especially in the early weeks, it's difficult to judge how many tasks to give because everyone has such a wide range of previous knowledge.

Exercise: Pythagoras Solver

Write a program which takes two floating-point numbers from the user which represent the lengths of the two shortest sides of a right-angled triangle. Your program should prompt appropriate messages like “Enter a float:” to let user know when to enter what type of values. The program should then calculate and print the length of the longest side.

Exercise: Typhoon Wind Strength

Using the JTWC Tropical Typhoon intensity scale, write a program which takes wind-speed as input and outputs the classification of intensity (eg, typhoon, violent typhoon, etc).

Submit your C program for this exercise to Moodle assignment.

Intensity Class	Wind speed (km/h)
Tropical Depression	<= 62
Typhoon	63 - 118
Strong Typhoon	119 - 156
Very Strong Typhoon	157 - 192
Violent Typhoon	>=193

Textbook 2.31 (Chapter 2 Exercise 31)

Table of squares and cubes.

Textbook 2.20

Converting seconds into hours, minutes, and seconds. Once you have successfully written this program, write a new program that does the opposite - read time in the format "hours:minutes:seconds" and converts that to seconds.

Hint: you can read in this input easily by asking `scanf` to read in literal characters that it will ignore. In the first argument to `scanf`, any character that is not preceeded by `%` will need to be present in the input the user types in for `scanf` to continue reading input. The first two paragraphs of textbook section 9.11.8 give an example (ignore the section after about assignment supression for now).

Textbook 2.30

Separating digits in an integer. This one is easy to do backwards but requires a little more thought to do forwards.

Textbook 2.29

A peek ahead of how the computer stores letters - as numeric codes. Appendix B of the book contains the ASCII code chart. Write the program in the question and compare your answers with the ASCII chart.

End