ITSC 2181 – Introduction to Computer Systems
Spring 2024
Module 03 – Unit 1: Lab

Objectives

- Practice how to write small C programs, including identifying and correcting `C` compiler error messages and warnings.
- Practice the use of `scanf` to read input (get data from the user).
- Practice the use of `printf` to format output (display data).
- Practice the use of `C` **expression**s to perform calculations and evaluate conditions in program logic.
- Practice the use of `if-else` flow of control statements a C program.

General Instructions

- Please do not write your email or user ID anywhere in the program. To identify your code, you may use your UNC Charlotte 800#

- In this lab, you will write a few short C programs. Each program needs to be in its own source code file, i.e., a file with the `.c` extension.

- **You need to test your code thoroughly before submitting it**.

- **To earn any credit, a program has to compile.** *You may comment-out lines that have errors to obtain partial credit for work done.*

- Programs need to compile cleanly to receive full credit, i.e., they do not produce any errors or warnings.

- It is essential that you **do your own work.**

    - **Do not use any external resources** (Internet, AI, friends, etc.). All cases of cheating will be taken very seriously.

    - **If you need help, please ask the instructor, TA/IA or CCI Tutoring center.**

Program 1 (20 points)

Write a program to calculate the **volume** of a cylinder, as follows:

1. The code uses the standard formula, $\boldsymbol{V = \pi r^2 h}$
   where ***r*** *is the radius and* ***h*** *is the height.*
   Also, you may use `3.14159` *for the value of PI*
2. The program asks the user to enter the radius and height of the cylinder and stores them into an appropriate data type.
3. The program uses floating point math and displays the result with two decimal digits of precision. A sample execution is shown below:

```
Enter radius: 10.5
Enter height: 3
Volume: 1039.08
```

4. Name your program `volume_of_cylinder.c`

Hints:

- Because C does not have an exponentiation operator, you will need to multiply *r* by itself twice to compute r^2
- For the value of π you should define a constant; however, this value can be hardcoded to `3.14159`. We will study later how to get C to use a "proper" value for π

Program 2 (40 points)

Using the tax rates table provided below. Write a program named `calculate_tax_amount.c` that does the following:

1. Asks the user to enter their income. You may assume that the user enters an integer value.
2. Calculates their tax due based on the table. The result must be a floating-point number.
3. Displays the tax amount, also a floating-point number. Format the output to two (2) decimal places.
4. Asks the user for the number of dependents. You may assume that the user enters an integer value. The user will enter zero if the answer is none.

5. Calculates and displays a tax credit of $450 per dependent, up to five dependents, i.e., the maximum credit is $2,250.
6. Displays the adjusted tax amount (tax minus credit). Format the output to two (2) decimal places.

Tax Rate	Income Bracket
12%	$0 to $9,275
17%	$9,275 to $37,650
27%	$37,650 to $91,150
30%	$91,150 to $190,150
35%	$190,150 and above

A few sample runs are provided below:

```
Enter income: 9200
Tax due = $1104.00
Enter the number of dependents (0 for none): 0
Tax credit = $0
Adjusted Tax = $1104.00
```

```
Enter income: 20000
Tax due = $3400.00
Enter the number of dependents (0 for none): 2
Tax credit = $900
Adjusted Tax = $2500.00
```

```
Enter income: 191025
Tax due = $66858.75
Enter the number of dependents (0 for none): 9
Tax credit = $2250
Adjusted Tax = $64608.75
```

Program 3 (40 points)

Write a program, named **`weather.c`**, that does the following:

1. Reads 3 temperatures from the console (standard input). Each value represents the high temp for a given day in a 3-day period.
2. Keeps track of the highest (max) temperature entered by the user.
3. Displays the day when the highest (max) temperature was recorded. If two or more days tie for the highest temperature, the program will report the last day with the highest temperature.

4. You do **NOT** need to loops in your implementation.
5. You do **NOT** need to use arrays for this program.

Hint: To save time, create a text file with test data and use input redirection when testing your code.

Two sample runs are provided below:

```
You will be asked to enter the daily high temperature for 3
consecutive days.
Enter a high temperature: 91
Enter a high temperature: 97
Enter a high temperature: 96

The highest recorded temperature in the 3-day period was: 97
degrees
Recorded on day #2
```

```
You will be asked to enter the daily high temperature for 3
consecutive days.
Enter a high temperature: 96
Enter a high temperature: 99
Enter a high temperature: 101

The highest recorded temperature in the 3-day period was: 101
degrees
Recorded on day #3
```

Submission Instructions

1. Create a folder (directory) on your computer.

2. Name the folder **ITSC_2181_M03_U1_Lab_*student-id*** Replace *student-id* with your UNCC student ID (800#), e.g. *8001231234*

3. Download and copy your program (source code) files into this folder.

4. Compress (Zip) the folder with all its contents. You should consider keeping the files for every lab in a separate folder (directory) from your other course materials. You can then use the *Send to, Compressed Folder* command on Windows or the *Compress* command on a Mac to create the Zip file.

5. Submit a single Zip file (created in #4, above) via *Canvas*.

Grading Rubric

- This lab is worth a total of **100 points**.
- **Do your own work. Do not use any external resources** (Internet, friends, etc.). All cases of cheating will be taken very seriously. **If you need help, please ask the instructor, TA/IA or CCI Tutoring center.**
- Programs need to compile to earn any credit. *You may comment-out lines that have errors to obtain partial credit for work done.*
- Your work will be graded on three (3) major components: Logic and flow of program, output, and formatting/organization. Refer to the following table for details.

Logic and Flow of Program - 60%	
Fully Correct and code compiles without errors.	Full Credit
Minor Errors and code compiles without errors OR some required functionality is missing.	75% Credit
Major Errors and/or code compiles with warnings OR significant portions of the required functionality are missing.	50% Credit
Completely incorrect/missing/does not compile.	No Credit
Output - 30%	
Fully Correct and matches formatting and layout of sample output provided.	Full Credit
Minor Errors.	75% Credit
Major Errors.	50% Credit
Completely incorrect output.	No credit
Formatting/Organization of Code - 10%	
Code is clear, easy to read and formatter according to the guidelines. Whitespace has been used appropriately, including indentation and blank lines. Comments are used when needed.	Full Credit
Needs minor improvement.	75% Credit
Needs major improvement.	50% Credit
No formatting/organization at all.	No Credit

Additional Deductions

- Code that produces warnings: -10% per program

- Incorrectly named files: -2 points per file
- *Students who cheat on <u>any</u> course assignment, lab, test or other activity will have their course grade reduced by one letter grade, regardless of the activity's point value, if it is their first offense at UNC Charlotte.*