CptS 121 - Program Design and Development

Lab 1: Introduction to Microsoft Visual Studio Community 2022 and Problem Solving with C

Assigned: Week of August 25, 2025
Due: At the end of the lab session

I. Learner Objectives:

At the conclusion of this programming assignment, participants should be able to:

- Analyze a basic set of requirements for a problem
- Declare variables
- Apply C data types and associated mathematical operators
- Comment a program according to class standards
- Logically order sequential C statements to solve small problems
- Compose a small C language program
- Compile a C program using Microsoft Visual Studio Community 2022
- Execute a program
- Create basic test cases for a program

II. Prerequisites:

Before starting this programming assignment, participants should be able to:

- Access Microsoft Visual Studio Community 2022 Integrated Development Environment (IDE)
- Apply basic problem-solving strategies
- Summarize key language elements of C including:
    1. Preprocessor directives
    2. Standard and user-defined identifiers
    3. Comments
    4. Punctuation
    5. Operators

III. Overview & Requirements:

Welcome to CptS 121 – Program Design and Development's first lab! Labs are constructed to be hands-on. So be ready to get your hands dirty! ***For this lab you should NOT use generative AI***.

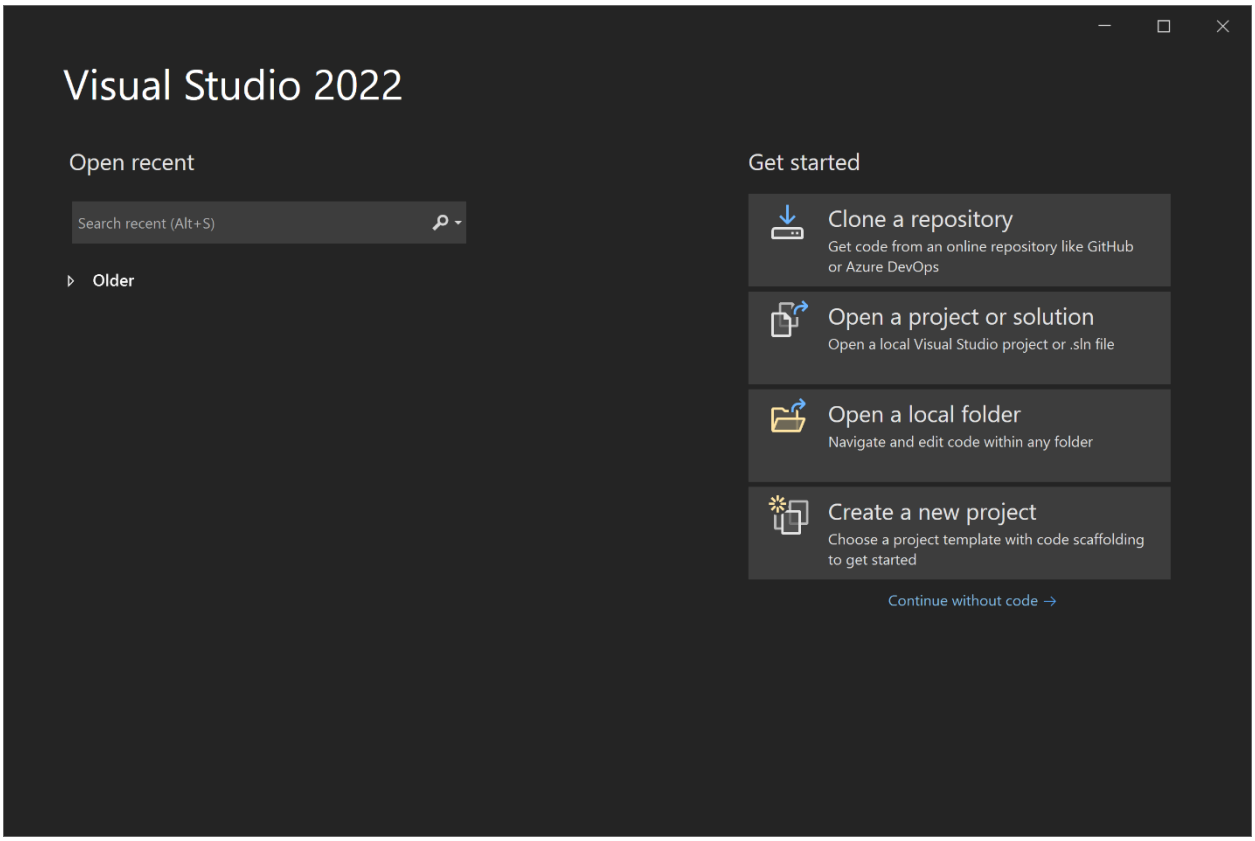
This lab, along with your TA, will help you navigate through Microsoft Visual Studio Community 2022 Integrated Development Environment (IDE). You will learn how to setup a console application starting from an empty project. You will take advantage of the text editor, compiler, linker, and loader that is built into Visual Studio to construct, execute, and test small C programs that solve basic mathematical problems.

Labs are held in a “closed” environment such that you may ask your TA questions. Please use your TAs knowledge to your advantage. You are required to move at the pace set forth by your TA. Please help other students in need when you are finished with a task. You must work in teams. However, I encourage you to compose your own solution to each problem. Have a great time! Labs are a vital part to your education in CptS 121 so work diligently.

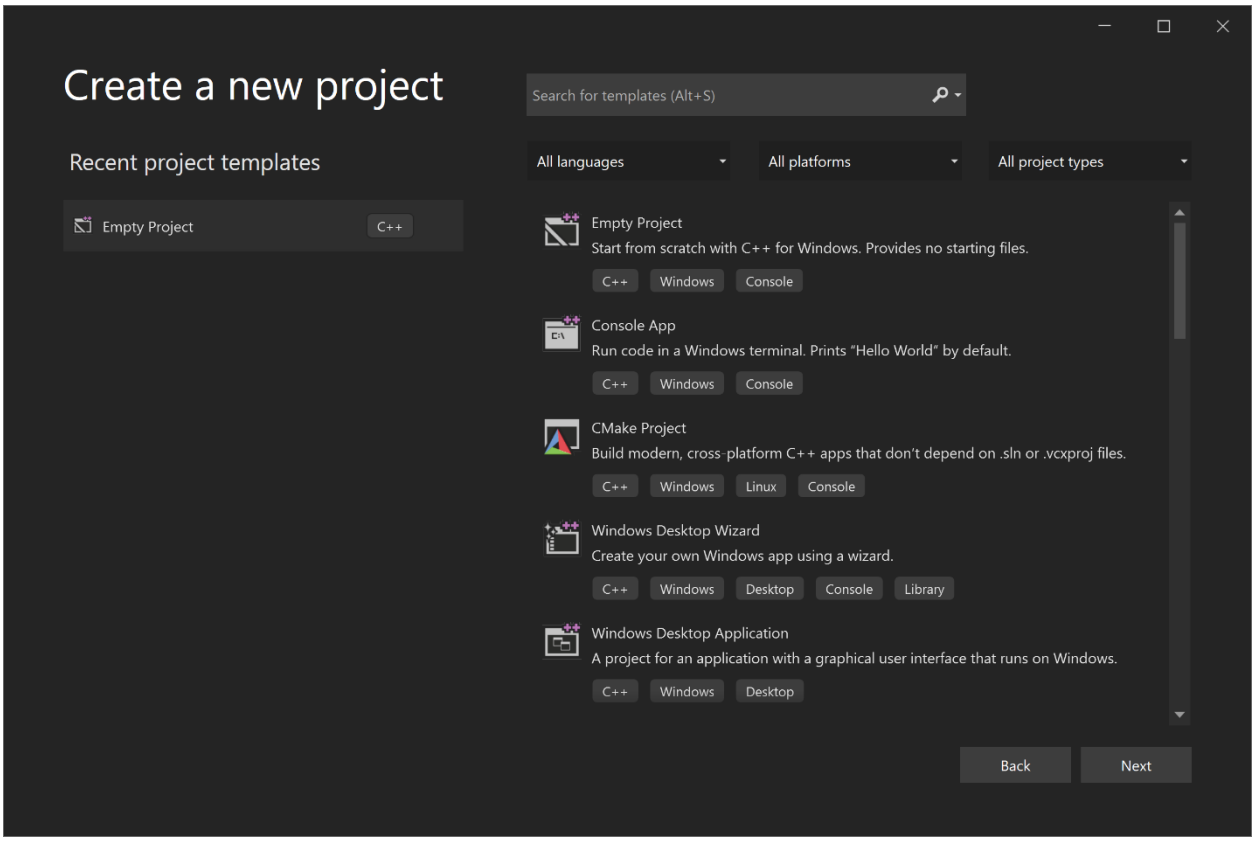
Task 1: Develop a small hello world program utilizing the Visual Studio development environment. All of the applications that we will build in this class will be Console applications. Console applications run in a command window, and have purely textual input and output.

Perform the following steps to build an initially empty console application:

1. Create a folder in an appropriate drive on your computer, for this class, named cpts121. This is where all programs that you create for this class will be saved.
2. From the start menu select Visual Studio 2022 (or equivalently click on the Visual Studio 2022 icon on the desktop if one exists). This will start the visual studio integrated development environment (IDE). The window shown below will load:

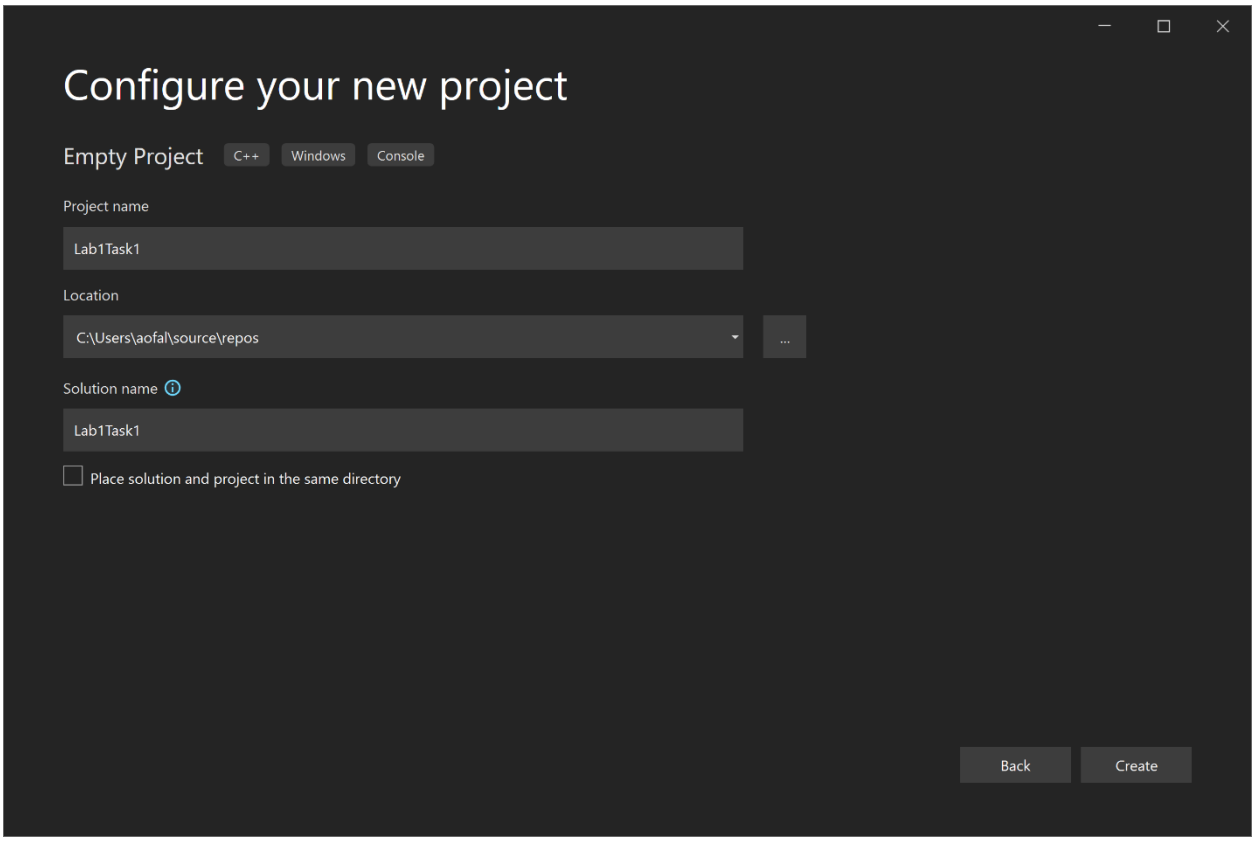


Select the “Create a new project” option. This will bring up the window shown below.

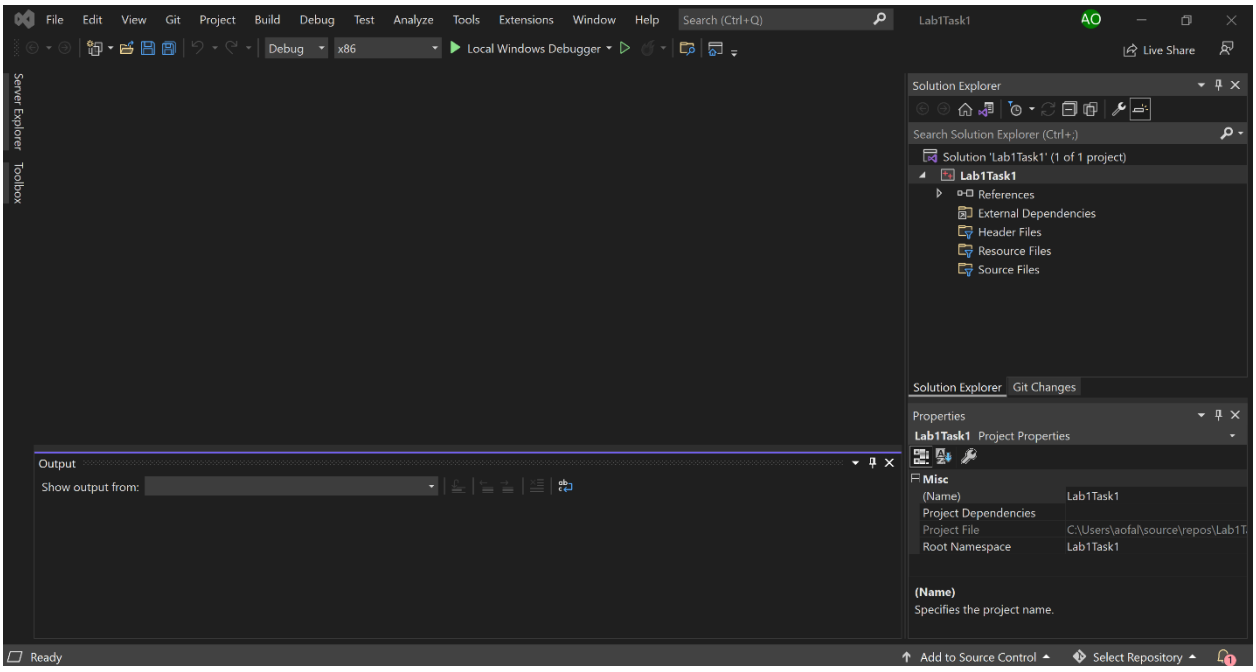


In this window, select “Empty Project” (see the shaded entry above). Make sure the project is for “C++”, although we will be using C. If you do not have these project templates, then you may NOT have installed the correct Visual Studio workloads, and will need to rerun the installer (be sure to select C++ desktop tools). Click “Next”.

Next, give the project a name as shown in the image below. A naming scheme will make it easier to find different solutions and projects later. At this time, name this project “Lab1Task1”. You also need to enter the Location; it should be in a location that is easy to find. Click “Create”.

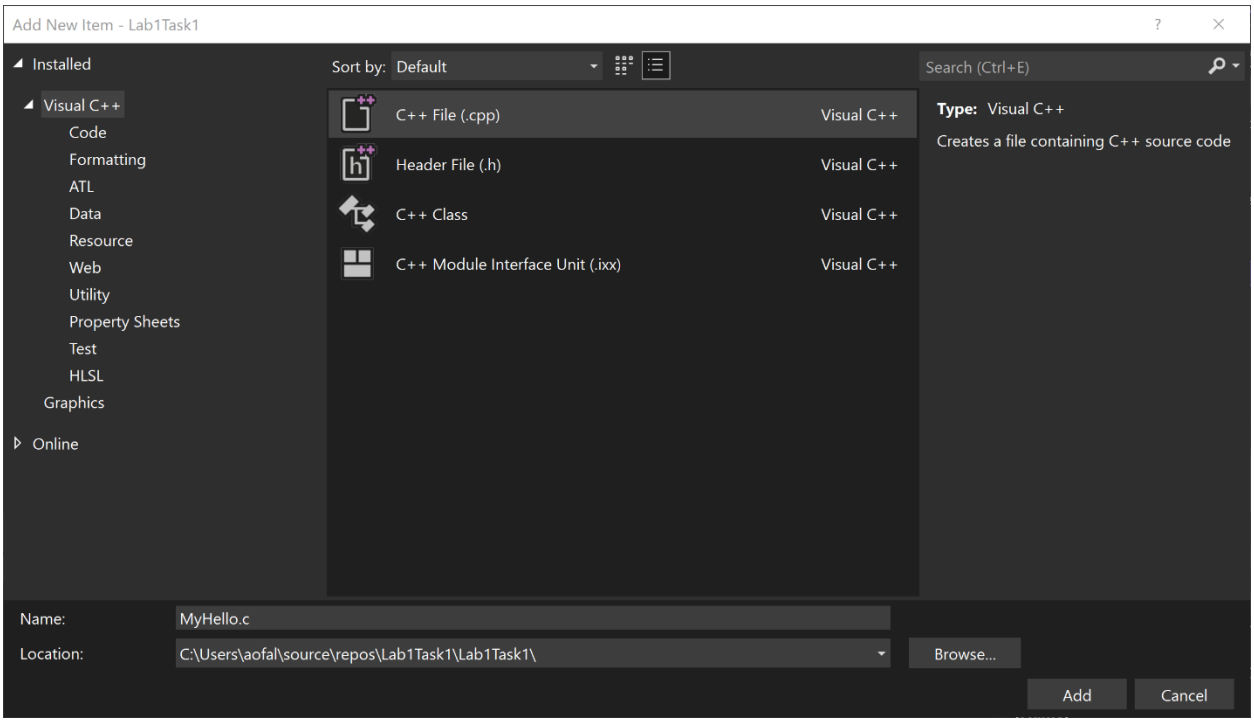


You will now see the editor load as shown below.

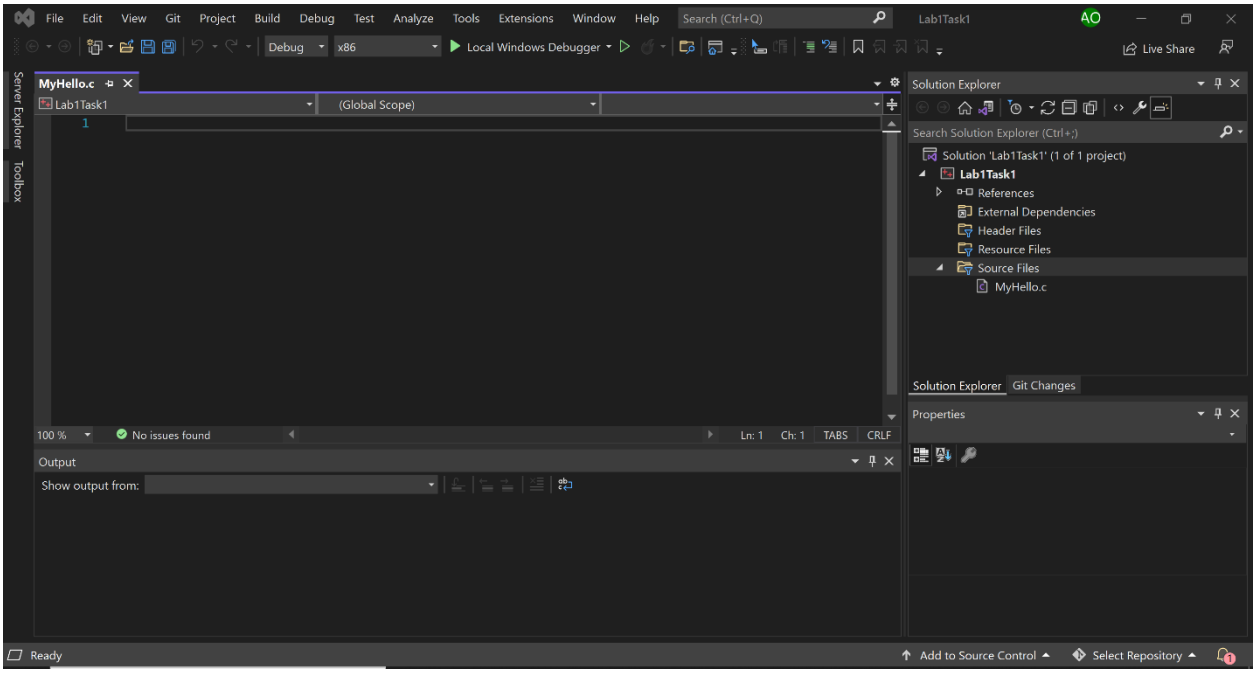


The IDE Window has changed a little bit now. There is information on the right-hand side of the window in the Solution Explorer. You now need to create a new file for your program. Do this by right clicking on the “Source Files” folder in the “Solution

Explorer", in the Visual Studio IDE, and selecting "Add" --> "New Item". Give your file a name in the filename text field as shown in the image below. Name the file MyHello.c. Make sure that you give your file the .c extension indicating that you would like the C compiler to be invoked once you compile your program. If you do ***not*** use the .c extension, the source file will default to a .cpp extension. This will cause the C++ compiler to be invoked. Next click "Add".



When you select "Add", you will see a file edit window come up inside the IDE with the name MyHello.c. You can enter your Hello World program here.



Make sure that for every program that you write, you have the following comment header block at the beginning of your source code file:

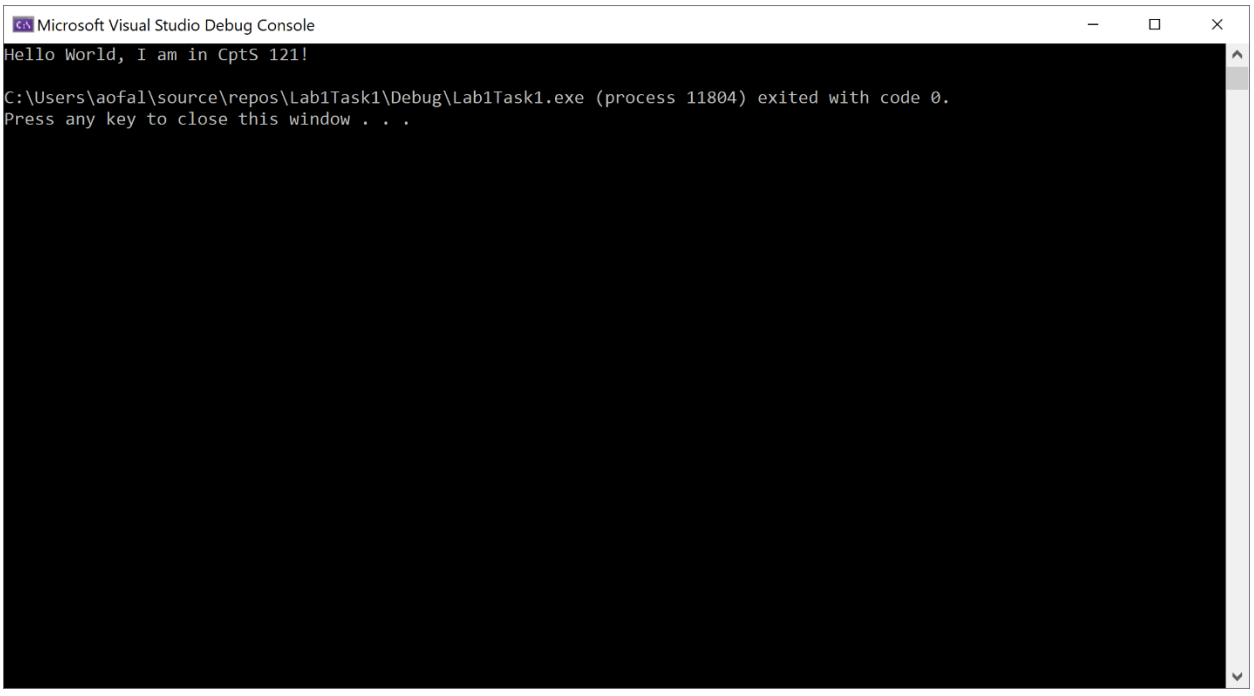
```
/*************************************************************
* Programmer: Your Name
* Class: CptS 121, Fall 2025; Lab Section X
* Programming Assignment: Lab1Task1
* Date: August 25, 2025
* Description: This program prints out a simple
*                "hello world" message.
*************************************************************/
```

For this first program you will only need one executable C statement to output a message. Your program should output "Hello World, I am in CptS 121!" with a printf ( ) statement.

Once you have entered the code for your simple hello world program you will build it from the Solution Explorer. Place the mouse pointer over Lab1Task1 in the solution explorer. Right click and choose “Build”. If you correctly entered the program source code you should see a "Build succeeded" message in the Build Window. If you had an error, you will have a failed message in the Build window and a fatal error in the Task List Window. Correct your mistakes and try again.

To run your program when it has successfully built, hit Ctrl-F5 or run it from the “Debug” tab in the IDE.

Life does not get much better than the joy experienced once you see the window below!



When you have successfully ran your hello world program, show your TA. Once you have completed this you may close this workspace under the file menu. Close this solution and create a new one for the next task. You should be able to move on to the rest of tasks when your TA instructs you to do so!

Before you start the next task, if you have not done so already, please be sure to team up with 2 or 3 other students!!! During all labs, I would like for you and your team to use the whiteboards as much as possible to illustrate ideas and algorithms. You should not start implementing code until you have flushed out most of your ideas as a team! Please be aware that most teams will have a diverse set of skills and I would like for you to encourage and support each other!

Task 2: Design, implement, compile, and test C solutions to the following problems. Once you have completed a problem, demonstrate your solution to your TA.

a. Write a program that prompts the user for two integer numbers (called number1_int and number2_int) and two floating-point numbers (called number1_float and number2_float). Please note that you must place the #define _CRT_SECURE_NO_WARNINGS statement, show below, at the top of your file. The code for declaring the variables and prompting the user is given below (please copy the code into your .c file):

```
#define _CRT_SECURE_NO_WARNINGS // necessary to ignore scanf_s () warnings

#include <stdio.h> // necessary to use printf () and scanf ()

int main(void) // the starting point for all C programs
{
	// we need to request memory for
	int number1_int = 0, number2_int = 0; // 2 variable declarations - reserves two memory
	blocks for integers and sets them to 0's
	double number1_float = 0.0, number2_float = 0.0; // reserves two memory blocks for
                                                // numbers with high precision (floating-point)

	printf("Enter two integer values: "); // prompt/ask the user
	scanf("%d%d", &number1_int, &number2_int); // read the integers typed through the
                                                // keyboard - %d is for "decimal", i.e. integers

	printf("Enter two floating-point values: "); // prompt/ask the user
	scanf("%lf%lf", &number1_float, &number2_float); // read the integers typed through
                                                // the keyboard - %lf is for "long float", i.e.
                                                // double precision numbers

	// place all code for the subtasks/operations below here




	return 0; // return a success status (value 0) indicating the program worked fine
} // end of the main () function
```

Perform the following operations on the numbers:

- Add number1_int and number2_int together and print the result as an integer value
- Subtract number1_float from number2_float and print the result as a floating-point value
- Multiply number1_int by number1_float and print the result as an integer value
- Divide number1_int by number2_int and print the result as an integer and a floating-point value (Ask yourself: What happens when you divide and integer by and integer? Does the result change because you print it as a floating-point number?)
- Divide number1_int by number2_float and print the result as an integer and a floating-point value (Ask yourself: How does this compare to the previous operations? We are now dealing with mixed data types.)
- Explicitly cast number1_int as a floating-point value and divide by number2_int. Print the result as a floating-point value.
- Try to mod number1_float by number2_float. Does the program compile? If not, fix it so that it does.
- Determine if number1_int and number2_int are even or odd. Print 0 if even and 1 if odd. You may not use "if" statements. Use the mod operator instead.

b. Write a program that prompts the user for inputs into Ohm's Law and determines the voltage. Print the voltage value. Ohm's Law: V = IR, where V is the voltage, I is the current, and R is the resistance of the circuit (all values are integers).

c. Write a program that prompts the user for inputs into Joule's Law and determines the power. Print the power value. Joule's Law: P = (V^2) / R, where P is the power, V is the voltage, and R is the resistance of the circuit (all values are integers). Do we have loss of precision? What kind of division is being performed?

d. Write a program that prompts the user for inputs into the third order polynomial equation provided and determines the y value. Print the y value. The third order polynomial equation is: $y = 3 * ax^3 + (1/4) * bx^2 + 10 * cx + (-5) * d$ (*x*, *y*, *a*, *b*, *c*, and *d* are integer numbers). Be careful here! What is (1 / 4) as an integer result? We do not want integer division in this case! How do we fix it? Do we ultimately have loss of precision?

e. Write a program that prompts the user for inputs into the circumference of circle equation provided and determines the circumference value. Print the circumference value. Circumference of a circle: circumference = 2 * PI * radius (*circumference* and *radius* are floating-point numbers; PI should be defined as a constant floating-point number and should not be entered by the user).

IV. Submitting Labs:

- You are not required to submit your lab solutions. However, you should keep them in a folder that you may continue to access throughout the semester.

V. Grading Guidelines:

- This lab is worth 10 points. Your lab grade is assigned based on completeness and effort. To receive full credit for the lab you must show up on time, work with a team, complete at least two-thirds of the problems, and stay until the TA has dismissed you. If you do not work with a team, then you will not earn any points for the lab.