CIS 246 – C++ Programming

Program:	**Week 01**
Points:	**20**
Chapter(s):	**1**
File(s) to Submit:	**Homework1.cpp**

Description

Write a C++ program using the Dev-Cpp IDE that prompts the user for some data, saves the data, and prints the data.

Instructions

Open Dev-Cpp and verify some settings:

1. Tools > Compiler Options
2. At the very top, verify that the following is visible in the "Compiler set to configure box"
   TDM-GCC 4.9.2 64-bit Release
3. There are two boxes visible below the "General" tab:
   a. Check the box "Add the following commands when calling the compiler"
   b. Add this text to the middle box: **-std=c++11**
   c. Check the box "Add the following commands when calling the linker"
   d. Add this text to the box at the bottom: **-static-libgcc**

We'll talk another time about what these settings mean. For now, just verify that they are set properly.

Create a New C++ Project:

1. File > New > Project (or click the first icon in the toolbar, and choose Project… from the dropdown)
2. Click "Console Application."
3. Verify that the radio button "C++ Project" is selected.
4. In the name box, you can use any name, or leave the default. You will not be submitting the project itself. You will be submitting the source code file that you create that is part of the project.
5. Choose a location to save this project and click Save.

Observe the file that has been created:

1. When you create a Console Application project in Dev-Cpp, a file called "main.cpp" is automatically added to the project and opened in the editor, with a main function started.

2. To see this file listed in the Project browser (the pane on the left), click the plus sign next to the name of the project.
3. Rename this file to “Homework1.cpp” by right-clicking on it in the Project browser > Rename.

Add a using directive:

1. Below the #include, and above main, add **using namespace std;** on a line by itself.
2. Click the Save icon (the floppy disk in the toolbar).
3. Be sure to save the file to the same location where you saved the project.

Declare an integer variable:

1. Within the main function, but above the return statement, declare an integer variable. The data type is int. Use any name you like.

Prompt the user:

1. Write the C++ statement that displays a prompt to the screen telling the user to type in a number.
2. There should not be a new line at the end of this statement.

Read what was typed:

1. Write the C++ statement that reads the input stream and saves the value in the integer variable you created above.
2. There **should not** be a new line at the end of this statement.

Display the number:

1. Write only one C++ statement that displays “The number you typed is” followed by the value the user typed. Be sure there is at least one space between the text and the number.
2. There **should** be a new line at the end of this statement.

Compile and test:

1. To compile in Dev-Cpp, use the Execute menu, or press the F9 key, or click the icon in the toolbar that looks like four squares in color.
2. After compilation is successful, run the program via the Execute menu, or by pressing the F10 key, or by clicking the icon that looks like a white square to the right of the Compile icon.
3. You can Compile and Run in one step via the Execute menu, or by pressing the F11 key, or by clicking the icon that looks like a square with four colors in it, to the right of the Run icon.

Once the program is working, you can submit it.

To Submit:

1. Close Dev-Cpp and save anything if prompted.
2. Locate the Homework1.cpp file on disk. Right-click and Send To… Compressed Folder. Use the default name for the zip or name it whatever you like.
3. **Submit only the zipped file** to the dropbox on Canvas. Do not submit any other file except Homework1.cpp, within a zip.

Clean Up:

When you create and use a project in Dev-Cpp, several files are created along with any source code files.

Files to keep are:

- The project file, which contains the settings related to this project. This file will use the name of the project, with the .dev extension.
- Any source code files (file extensions .cpp and later, .h)

For example, if you named this project Project1, and you named the source code file Homework1.cpp, then keep these two files:

Project1.dev

Homework1.cpp

Keep them together in the same folder. It doesn't have to be the same folder where you originally created them, but they have to stay together.

All other files that may have been created, such as .exe, .o, .layout, and Makefile.win, can be left as they are, or discarded. If you delete them, they will be automatically recreated when you open the project again.

To open the project again, either use the File > Open menu from within Dev-Cpp, OR double-click on the .dev file, which will open Dev-Cpp automatically, and open the project in it.

General Requirements

For complete credit, you must:

1. **MEET ALL REQUIREMENTS ACCORDING TO THE INSTRUCTIONS** – Follow the instructions as written for completing this project, even if you [think you] know a "better" way to do something. I will deduct points for not following instructions as written.

2. **INCLUDE COMMENTS** – Include comments in your code. There must be a comment at the top of each source code file that includes your name, the program number, and a description of the class. There must be comments at each important step in your algorithm that describes that step.
3. **FOLLOW BEST PRACTICES** – Follow best practices in C++ programming, including, but not limited to, appropriate use of private/public, appropriate use of classes and/or header files, sets & gets, white space, alignment, meaningful variable names, naming conventions, using directives, etc. Points will be deducted for sloppy code that is hard to read, even if it works, so pay attention to these details.
4. **SUBMIT ALL FILES BEFORE THE DUE DATE** – Submit a .zip of ONLY source code files to the dropbox for this assignment on Canvas before the due date. Do not submit anything except .cpp and/or .h, within a zip. Do not submit .exe files. Do not submit project files from an IDE.

Sample Run

Note that the user should be able to type at the end of the same line as the prompt, NOT on the next line below.

```
Type in a number: 10
The number you typed is 10
```