Assignment 1
C/C++ Programming I

C1A1 General Information

Course Assignment/Exercise Notation Conventions: Each weekly "assignment" consists of several "exercises". Throughout this course I commonly refer to these using an abbreviated notation, where a form like **C1A2E3** would refer to exercise 3 in assignment 2 of the "C/C++ Programming I" course and **C1A2** would refer to the entirety of assignment 2 of that course.

Getting Started: Before starting your first assignment you must have the appropriate tools for developing your software and the best way to get them is to download and install one of the many free integrated development environment (IDE) packages. These integrate the compiler, editor, and other necessary tools into a convenient GUI application. Although you are free to use any tools you wish and any operating system that will support them, I recommend Microsoft's "Visual Studio Community" for Windows, "Xcode" for Max OS X, and "Code::Blocks" for Linux. Information on obtaining, installing, and using them is available in the appropriate version of the course document titled "Using the Compiler's IDE...", a link to which is located on the "Assignments" page of the course Web site. I'm sorry but I don't have information on other IDE's or operating systems.

Source Code Files: Header Files and Implementation Files: "Source code" files contain the code necessary for a program to be built without errors and are divided into the two categories "header" files (`.h`, etc.) and "implementation" files (`.c`, `.cpp`, etc.). Not all programs require header files but at least one implementation file is always required. Header files are designed to be included in other files using the `#include` directive but implementation files are not. By placing items that might be needed by multiple other files in header files and including them in those files the bad practice of literally duplicating the needed items in each file can be avoided. Because of their multiple usages, however, header files must never contain anything that will result in an error if more than one file includes them. Files containing data that your program reads or writes are not considered source code files but are instead "data files".

Although some of the following terminology has not yet been discussed in this course it is placed here for completeness and for future reference: Header files typically contain things like macro definitions, inline function definitions, function prototypes, referencing declarations, typedefs, class/structure/union descriptions, and templates, although any of these that will only ever be needed by one specific implementation file may be placed in that file instead. Header files must not contain non-inline function definitions or defining variable declarations; these must be placed in implementation files instead. The header files that are supplied with a compiler provide the information it needs to properly compile code that uses the various functions, macros, and data types available in the compiler's libraries.

Exercise Submission Procedure: Get an exercise to work first on your computer, then submit it to the "Assignment Checker" and wait for the results to be returned. If there are any errors or warnings make the appropriate corrections and resubmit, repeating as necessary until all issues are corrected. Additional details are provided in each exercise and in the course document titled "Preparing and Submitting Your Assignments".

Get a Consolidated Assignment 1 Report (optional)

If you would like to receive a consolidated report containing the results of the most recent version of each exercise submitted for this assignment, send an empty email to the assignment checker with the subject line **C1A1_*ID***, where ***ID*** is your 9-character UCSD student ID. Inspect the report carefully since it is what I will be grading. You may resubmit exercises and report requests as many times as you wish before the assignment deadline.

C1A1E0 *(6 points total - 1 point per question – No program required)*

Assume language standards compliance and any necessary support code unless stated otherwise. Testing erroneous or implementation dependent code by running it can be misleading. These are not trick questions and each has only one correct answer. Major applicable course book notes are listed.

1. Which of the following is a character literal? (Note 1.5)
   A. '\A\'
   B. 1 single quote between 2 double quotes
   C. '\'
   D. '\xd'
   E. 1 single quote between 2 single quotes

2. Of the following
   main Main int _ scanf goto 85Text
   which are legal identifiers? (Note 1.4)
   A. all of them
   B. only **Main** and _
   C. only **main**, **Main**, _, and **scanf**
   D. only **85Text**
   E. only **Main**, _, and **85Text**

3. Assuming the ASCII character set, if variable *temp* is of the correct type and the user enters 5 spaces followed by ***3.5*** what value ends up in *temp* for the following code:
   scanf("\t%c", &temp);
   (Notes 1.13 and B.1)
   A. the ASCII value of the tab character
   B. an octal value of 63
   C. the actual numeric value ***3.5***
   D. a hexadecimal value of 20
   E. the ASCII values of 5 *space* characters followed by the ASCII values of the three characters ***3.5***

4. Which is the most appropriate header file to include in a C++ program if the macro *EXIT_SUCCESS* is to be used? (Note 1.9)
   A. iostream
   B. cstdio
   C. stdlib.h
   D. cstdlib
   E. stdio.h

5. Given the declarations
   char c;
   short s;
   int i;
   double d;
   float f;
   which of the following uses a variable of the wrong data type? (Note 1.13)
   A. scanf("%c", &c)
   B. scanf("%s", &s)
   C. scanf("%d", &i)
   D. scanf("%f", &f)
   E. scanf("%lg", &d)

6. Predict the output from:
   int x = 7;
   cout << ++x;
   printf("%i", x--);
   printf("%f", x);
   (Notes 1.7, 1.11, and 1.12)
   A. 88 *followed by garbage*
   B. 887.000000
   C. 787
   D. 877
   E. *788*

Submitting your solution

Using the format below place your answers in a plain text file named **C1A1E0_Quiz.txt** and send it to the Assignment Checker with the subject line **C1A1E0_*ID***, where ***ID*** is your 9-character UCSD student ID.

-- Place an appropriate "Title Block" here --
1. A
2. C
etc.

See the course document titled "Preparing and Submitting Your Assignments" for additional exercise formatting, submission, and Assignment Checker requirements.

C1A1E1 *(7 points – C++ Program)*

Exclude any existing source code files that may already be in your IDE project and add a new one, naming it **C1A1E1_main.cpp**. Write your program in this file.

26 of the 128 characters in the ASCII character set can only be represented by specifying their underlying numeric values, while the remaining 102 characters can be represented literally or as simple escape sequences, as appropriate. Numeric representations, except for the null character, are cryptic and non-portable and are a very poor programming practice if a literal or simple escape representation is available. In this exercise, however, you will represent some characters numerically just to become familiar with the technique.

When displayed using `cout` all three string literals below will produce the same output except for the first character, as indicated in the adjacent comments. The first uses only literal characters and a simple escape sequence, the second uses only octal escape sequences, and the third uses only hexadecimal escape sequences. Note the cryptic and non-portable nature of the second two:

```
"1. Hello\n"                                  // "1. Hello\n"
"\62\56\40\110\145\154\154\157\12"            // "2. Hello\n"
"\x33\x2e\x20\x48\x65\x6c\x6c\x6f\xa"         // "3. Hello\n"
```

Write a program that uses a single `cout` to display the strings represented by 4 string literals. The first and last string literal must not contain any numeric escape sequences, the second must contain only octal escape sequences, and the third must contain only hexadecimal escape sequences. The output from your program must be the following, including the 1-4 line numbering:

```
1. C/C++ (,;{=$?)
2. C/C++ (,;{=$?)
3. C/C++ (,;{=$?)
4. \"\\%\n33+
```

- Do not output any information other than that shown above;
- Do not use any variables;
- Do not use `cout` more than once;
- Do not place more than one string literal on the same line of code.

Submitting your solution

Send your source code file to the Assignment Checker with the subject line **C1A1E1_*ID***, where ***ID*** is your 9-character UCSD student ID.

See the course document titled "Preparing and Submitting Your Assignments" for additional exercise formatting, submission, and Assignment Checker requirements.

Hints:
See notes 1.5 and B.1.

C1A1E2 *(7 points – C Program)*

Exclude any existing source code files that may already be in your IDE project and add a new one, naming it **C1A1E2_main.c**. Write a program in that file to display the exact text below using `printf`:

```
Commenting code "improperly" can lead to a career in fast food.
"Portable" code runs the same way on all platforms.
\n produces \n, \r\n, or \r, depending upon the OS.
%d is for char/short/int in printf but only for int in scanf.
Don't use %le, %lf, or %lg for type double in printf.
Use %% in a printf format string to display %.
```

Your program must:

1. not call `printf` more than once;
2. not use the underlying numeric value of any character;
3. not use the `%c`, `%s`, or `%[]` conversion specifications.

Submitting your solution

Send your source code file to the Assignment Checker with the subject line **C1A1E2_*ID***, where ***ID*** is your 9-character UCSD student ID.

See the course document titled "Preparing and Submitting Your Assignments" for additional exercise formatting, submission, and Assignment Checker requirements.

Hints:

To display a percent character located within a `printf` control string (the control string is the first argument of `printf`) use two percent characters together. To represent a backslash character in any string use two backslash characters together. The compiler automatically concatenates multiple string literals separated only by zero or more whitespaces into one string, including string literals on separate lines.