Assignment 1
C/C++ Programming I

General Information

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.

Too Easy?

If, after completing the three exercises in this document, you would like to gain more programming experience please contact the instructor at MeanOldTeacher@MeanOldTeacher.com and request a supplemental set of Assignment 1 exercises. Note, however, that these are only for your own benefit and you will not receive any extra credit for doing them.

Exercise 0 (6 points – 1 point per question – No program required)

Language standards compliance and appropriate header file inclusion is assumed. Testing code by running it is sometimes misleading due to implementation dependence. These are not trick questions and there is only one correct answer to each. Applicable notes from the course book are listed.

1. Which of the following is not a character literal?
   (Note 1.5)
   A. 'A'
   B. 1 double quote between 2 single quotes
   C. '\0'
   D. '\x5'
   [✓ E.] 1 single quote between 2 single quotes

2. Which of the following is not a string literal?
   (Note 1.5)
   A. nothing between 2 double quotes
   B. spaces between 2 double quotes
   C. 1 single quote between 2 double quotes
   [✓ D.] 1 double quote between 2 double quotes
   E. \t between 2 double quotes

3. Assuming the ASCII character set, predict the output from:
   printf("\x49\146\155\155\x70\x0021");
   (Note 1.5 & Note B.1 ASCII Code Chart)
   A. nothing - The compiler will not allow this.
   B. \x49\146\155\155\x70\x0021
   C. \xe\1)(
   [✓ D.] Ifmmp!
   E. none of the above

4. What data types are acceptable for x in the expression *printf("%d", x)*
   (Note 1.11)
   A. **int** only
   B. **int** and **double** only
   [✓ C.] **char**, **short**, and **int** only
   D. **char**, **short**, **int**, and **long** only
   E. any of the above

5. What data types are acceptable for x in the expression *scanf("%d", &x)*
   (Note 1.13)
   [✓ A.] **int** only
   B. **int** and **double** only
   C. **char**, **short**, and **int** only
   D. **char**, **short**, **int**, and **long** only
   E. any of the above

6. For the declaration ***char*** *ch;* expressions that will input a character from the user into the variable *ch*, skipping all leading whitespace are:
   (Note 1.15)
   A. scanf("%c", &ch) *and* cin >> ch
   B. ch = getchar() *and* cin.get()
   C. scanf("%c", &ch) *and* cin >> &ch
   [✓ D.] scanf("\n%c", &ch) *and* cin >> ch
   E. scanf("%c", ch) *and* cin >> (**char**)&ch

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.

Exercise 1 *(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.c**. Write a program in that file to display the exact text below using `printf`:

```
In C/C++ the case of letters is significant.
main is where program execution begins.
A semicolon terminates most statements.
10% of "nothing" is 100% of "nothing".
Use \n to cause a newline; use \t to cause a tab.
Use \a to cause a beep (only on some platforms)!
```

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 **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:
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 character literals separated only by zero or more whitespaces into one string, including string literals on separate lines.

Exercise 2 (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.cpp**. Write a program in that file to display a value in decimal, octal, and hexadecimal. Your program must:

1. not use `cout` more than twice;
2. declare one type **int** variable;
3. use `cout` to prompt the user to enter any decimal integer numeric value;
4. use `cin` obtain and store the user input value in the type **int** variable;
5. use `cout` to display the input value in decimal, octal, and hexadecimal using the following format, where **D** represents the decimal value, **O** represents the octal value, and **H** represents the hexadecimal value:
   ```
   D decimal = O octal = H hexadecimal
   ```

For example, if the user were to enter **22** your program would display the following, all on one line:

```
22 decimal = 26 octal = 16 hexadecimal
```

Manually re-run your program several times, testing with at least the following 5 input values:

7 8 15 16 -1

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:
See note 1.12. Use the `hex`, `oct`, and `dec` manipulators to change the integer output format to hexadecimal, octal, or decimal, respectively. The selected format will remain in effect until explicitly changed. Upon program startup the format is always decimal.

Get a Consolidated Assignment 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.