Written Assignment 1
Variables, Data Types and Math

ECE 131 – Programming Fundamentals
Instructor: M. Wolverton

W1.1 Definitions

Define each of the following items in the context of C.

Name	Description
Source Code	
Compiler	
Variable	
Operator	
Control Structure	
Function	
Directive	

W1.2 Operators

Describe the following symbols in the context of C source code.

Symbol	Name	Description
=		
+ - * /		
%		
++ --		
+= -= *= /=		
#		

W1.3 Exceeding Maximum or Minimum Values

Consider the following block of C code.

```
int a = 2000000000; // 2 billion, 2x10^9
int b = 1000000000; // 1 billion, 1x10^9
int c = a + b;
```

What int value does c hold at the end of this code block?

Attempt this calculation in a graphing calculator (e.g. www.desmos.com). Explain why you obtain different results.

W1.4 Data Type Properties

Experiment using `sizeof()`, `printf()` and arithmetic operations to determine the following data type information information for our classroom workstation C compiler. Note that `sizeof()` returns the data size in units of 'char size'. To convert to bits multiply by `CHAR_BIT` and `#include <limits.h>`.

Integer Data Types					
Name	**Width (bits)**	**Min. (signed)**	**Max. (signed)**	**Min. (unsigned)**	**Max. (unsigned)**
char					
short					
int					
long					

Floating Point Types				
Name	**Width (bits)**	**Significant Digits (base 10)**	**Largest Magnitude (power of 10)**	**Smallest Magnitude (power of 10)**
float				
double				

W1.5 Changing Data Types

Consider the following block of C code.

```
char ch;
int i;
i = 321;
ch = i;
printf("%c", ch);
```

What char value (character) does ch hold at the end of this code block? Explain why that letter is the result.

W1.6 Changing Data Types - II

Consider the following block of C code.

```
char ch;
int i;
ch = 'q';
i = ch;
printf("%d", i);
```

What int value (number) does i hold at the end of this code block? Explain why that number is the result.

W1.7 Arithmetic Errors and Data Types

Suppose a variable, score, has a value between 0 and 20. The following different code options are being considered for calculating score's percent out of 20.

Option A:

```
int score = 18;
score = (score / 20) * 100;
printf("%d", score);
```

Option B:

```
int score = 18;
score = ((double)score / 20) * 100;
printf("%d", score);
```

Option C:

```
int score = 18;
score = (score / 20.0) * 100;
printf("%d", score);
```

Option D:

```
int score = 18;
score = (int)(score / 20.0) * 100;
printf("%d", score);
```

This code should produce 90 (18/20 = 90%), but some options do not work as intended. What does each result produce? Why do the options that don't work correctly produce the results you see?