TCSS333
C for System Programming
Programming Assignment 1

DUE: See Canvas Programming Assignment 1 link.

You are to submit a single .c file to Canvas and nothing else (do not zip anything, no Eclipse projects, etc.).

Please note: All programs in this course must run correctly in a Windows environment under Cygwin. If you are using a Mac, make sure your program runs correctly on a Windows based machine or using Linux on the lab computers as demonstrated in class.

Write an interactive program that performs addition, subtraction, multiplication, and division of binary numbers entered by the user. The binary numbers will include an integer and fractional part: 101.01 = 5.25
The entered numbers should always include a decimal point and at least 1 digit on each side, i.e. 6 is 110.0, 0.375 is 0.011, etc. The input will always be in the form: operand operator operand where the operands are binary numbers (as previously described) and an operator symbol of +, -, *, /. Once the expression is entered, the program will display a binary result. Termination occurs when the user simply type the letter q or Q. A sample session is as follows:

```
Enter an expression using binary numbers or Q to quit:  1101.001 * 101.1101
 = 1001100.0100101
Enter an expression using binary numbers or Q to quit:  0.0000000001 * 0.0000000001
 = 0.00000000000000000001
Enter an expression using binary numbers or Q to quit:  0.111 * 1000.0
 = 111.0
Enter an expression using binary numbers or Q to quit:  11.0 * 11.0
 = 1001.0
Enter an expression using binary numbers or Q to quit:  11.11 + 11.11
 = 111.1
Enter an expression using binary numbers or Q to quit:  101.1 / 10.0
 = 10.11
Enter an expression using binary numbers or Q to quit:  1001.11 - 11.01
 = 110.1
Enter an expression using binary numbers or Q to quit:  q
```

Do not consider negative numbers as they will not be tested.

To perform the operations, the binary numbers will need to be converted to double, perform the operation on the doubles, and then convert the double result into binary as you print it out.

If you choose to read one character at a time using scanf, please note: to read the next non-whitespace character, you need scanf(" %c", &myChar); (notice the blank before %c).

If you choose to read the two operands as strings (arrays of char) and the operator as char you would declare them as:

```
char bi1[50] = {'\0'};
char bi2[50] = {'\0'};
char op;
// . . .
```

and read the first operand by itself (need to check for just a ‘q’ or ‘Q’), check for ‘q’ and if not there, read the operator and second operand and complete the conversions/expression evaluation/results output:

```
   scanf(" %s", bi1);
   if   (bi1[0] != 'q' && bi1[0] != 'Q') {
      scanf(" %c %s", &op, bi2);
      // do your conversions here, etc.
```

You are expected to use functions for input, output of a binary number, and whatever other functions you feel are necessary (don't forget: function prototypes are listed before main which is followed by the function definitions)

Global variables are variables declared above or in between functions. You may **not use** global variables! If you want to move data from one function to another, use parameters and/or return values.

Algorithm to convert from binary to float: Suppose we're working with the binary number 110.011 Start with value = 0. Until you run into the binary point, repeat this procedure: - double the value - add the next bit (0 or 1) Doing this with 110 should give a result of 6.

Step over the binary point. Starting with 0.5 as the current power of 2, repeat this procedure: - if the next bit is a 1, add in the current power of 2 (for 0, don't) - divide the current power of 2 by 2 (0.5, 0.25, 0.125,...) Doing this to .011 would add .375 (0.25 + 0.125) to the value calculated above for a total of 6.375

Algorithm to convert from float to binary: Starting with 1, calculate a power of 2 that is greater than the value you're trying to convert, i.e. keep doubling the power of 2 number until it is greater than the float value. In the case of the double 30.25, that would be 32 (1, 2, 4, 8, 16, 32). Now divide that power by 2 (32 / 2 = 16 in this case). At this point, repeat the following procedure: - If you can subtract the current power of 2 (we're starting with 16 in the current example) from the double (currently is 30.25), do that and output a 1 - Otherwise output a 0. (Using our example, what this step means is, if 30.25 – 16 is >= 0.0, print a 1 or else a 0) - Divide the current power of 2 by 2 Do this as long as the current power of 2 is at least 1 (print at least one digit before the binary point). (For the value 30.25, we would now have printed out 11110 while substracting 16,8,4,2) Print out the binary point. Then start the same repetitive procedure again. (The current power of 2 starts at 0.5) This will lead to more 1's and 0's based on whether you can subtract the current power of 2. You must print at least one digit after the binary point. You can stop printing binary fraction digits when either the remaining value of the double is 0 or you have printed out 20 fractional digits. The following illustrates this process by showing the contents of the double (value), the power of 2 (pow2), and the resulting string (result):

value	pow2	result	
30.25	1		// Finding power of 2 > value
	2		
	4		
	8		
	16		
	32		// Here we are! Now, divide pow2 by 2
	16	1	// Now start finding bits: 30.25 – 16 >= 0.0 so add a '1'
14.25	8	1	// Subtract pow2(16) from value & divide pow2 by 2
		11	// 14.25 – 8 >= 0.0 so add a '1' to the string and…
6.25	4	11	// Subtract pow2(8) from value & divide pow2 by 2
		111	// 6.25 – 4 >= 0.0 so add a '1' to the string and…
2.25	2	111	// Subtract pow2(4) from value & divide pow2 by 2
		1111	// 2.25 – 2 >= 0.0 so add a '1' to the string and…
0.25	1	1111	// Subtract pow2(2) from value & divide pow2 by 2
		11110	// 0.25 – 1 is not >= 0.0 so add a '0' to the string and…
	0.5	11110	// Do NOT subtract-failed above test – divide pow2 by 2
Xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx Completed integer portion as pow2 is < 1.			
		11110**.**	// Note: pow2 is < 1. Done with integer portion & add decimal pt.
		11110.0	// 0.25 – 0.5 is not >= 0.0 so add a '0' to the string and…
0.25	0.25	11110.0	// Do NOT subtract-failed above test – divide pow2 by 2
		11110.01	// 0.25 – 0.25 is >= 0.0 so add a '1' to the string and…
0.0	0.125	11110.01	// subtract pow2(0.125) from value & divide pow2 by 2

Because value = 0.0, we are finished. If value was not equal to 0, we would continue until we filled 20 fractional digits in the result or stop if value becomes 0. If still unsure, stop by my office for more help.

NOTE: if the double (value) starts with a value less than 1.0, add a zero and decimal point to the result and start the above process for the fractional portion (pow2 should begin with 0.5 in this case).

Basic Code Quality Requirements (these apply to future assignments as well):

Your code must include your name in a comment at the top of the program!

Write multiple functions that each do one simply described thing. If it's not possible to see all the code of a function at once (without scrolling), the function is probably too long.

C requires you to define a function before you call it. If function A calls another function B, B should appear above A. Since main should appear before all other functions, add function prototypes before main .

The names you choose for variables, functions, etc. should be meaningful and descriptive.

Redundant code is code that is clearly repetitive and doesn't need to be. If you find yourself copying and pasting while coding, you very likely are writing some redundant code. Redundant code should be cleaned up before submission. Use functions and loops to eliminate redundancy.

Your code must be neatly and correctly indented. Don't use the tab key to indent. Tab doesn't have the same effect in every editor, so code that is beautifully indented in one editor can look bad in another. The text book is a good model for indenting.

For 10 points extra credit, write you code to 'remember' the binary input strings, output the double values and output the binary values as such:

```
Enter an expression using binary numbers or Q to quit: 1101.001 * 101.1101
13.1250000000 * 5.8125000000 = 76.2890625000
1101.001 * 101.1101 = 1001100.0100101
Enter an expression using binary numbers or Q to quit: 0.0000000001 * 0.0000000001
0.0009765625 * 0.0009765625 = 0.0000009537
0.0000000001 * 0.0000000001 = 0.0000000000000000001
Enter an expression using binary numbers or Q to quit: 0.111 * 1000.0
0.8750000000 * 8.0000000000 = 7.0000000000
0.111 * 1000.0 = 111.0
Enter an expression using binary numbers or Q to quit: 11.11 + 11.11
3.7500000000 + 3.7500000000 = 7.5000000000
11.11 + 11.11 = 111.1
Enter an expression using binary numbers or Q to quit: 101.1 / 10.0
5.5000000000 / 2.0000000000 = 2.7500000000
101.1 / 10.0 = 10.11
Enter an expression using binary numbers or Q to quit: 1101.11 - 101.01
13.7500000000 - 5.2500000000 = 8.5000000000
1101.11 - 101.01 = 1000.1
Enter an expression using binary numbers or Q to quit: q
```