CSCI 2271 Computer Systems
Assignment 1: Input / Output
Due Friday, January 25

Your assignment is to write three C programs. The purpose of these programs is to give you experience using C and to test your ability to effectively read input and print output in C. Please do not use "advanced" constructs that we have not covered in class, in particular arrays.

1. Write a program that asks the user to enter the numbers from 1 to 16 (in any order) and then displays the numbers in a 4 by 4 arrangement, followed by the sums of the rows, columns, and diagonals. For example:

```
adminuser@adminuser-VirtualBox ~/Desktop/classDemos $ ./a.out
Enter the numbers from 1 to 16 in any order: 16 3 2 13 5 10 11 8 9 6 7 12 4 15 14 1

16  3  2 13
 5 10 11  8
 9  6  7 12
 4 15 14  1

Row sums: 34 34 34 34
Column sums: 34 34 34 34
Diagonal sums: 34 34
adminuser@adminuser-VirtualBox ~/Desktop/classDemos $
```

If the row, column, and diagonal sums are all the same (as they are in this example), the numbers are said to form a ***magic square.*** The magic square shown here appears in a 1514 engraving by artist and mathematician Albert Durer. (Note that the middle numbers in the last row gives the date of the engraving.)

2. Modify **count.c** so that it also prints the total number of words in the input text and the average word length. Print the average word length with two digits to the right of the decimal point. Call your program **avgWordLength.c**. For the purpose of this assignment, assume that a word is a sequence of non-whitespace characters. The best way to test for a whitespace character is to use the library function **isspace**, which is in the library **<ctype.h>.** For example, the function call isspace(c) returns 1 if char c is whitespace, and 0 otherwise. For an example of the program in action:

```
adminuser@adminuser-VirtualBox ~/Desktop/classDemos $ gcc avgWordLength.c -o avgWordLength
adminuser@adminuser-VirtualBox ~/Desktop/classDemos $ ./avgWordLength
Enter text. Use an empty line to stop.
fgjglkk ofjvf
               abc

Your text has 2 lines and 20 characters.
Your text has 3 words, with an average length of 5.00.
adminuser@adminuser-VirtualBox ~/Desktop/classDemos $ avgWordLength <mobydick.txt
Enter text. Use an empty line to stop.
Your text has 23244 lines and 1256167 characters.
Your text has 214112 words, with an average length of 4.71.
adminuser@adminuser-VirtualBox ~/Desktop/classDemos $
```

3. Download the file Calculator.java, which is part of the assignment handout. This program simulates a 4-function calculator. Input to the calculator consists of a sequence of lines. The first character of each line is one of the following 7 operators: '+', '-', '*', '/', '=', 'C', and 'Q'. The first 4 operators are arithmetic operators. Lines having one of these operators also contain a floating-point number (possibly separated by whitespace) following the operator. The calculator always has a "current amount", which is initially 0. The meaning of an arithmetic operation is to modify the current amount by performing the specified arithmetic to it. The remaining 3 operators are the only things on their lines. Operator '=' causes the current amount of the calculator to be printed. Operator 'C' causes the current amount to be 0. Operator 'Q' causes the program to quit.

   Your assignment is to translate this program into C. You must keep the code as close to the Java version as possible, changing something only when necessary. In particular, don't get rid of (or add) any functions. But don't forget to add declarations for these functions to your C code!

   The only tricky part concerns input. Each line consists of the following: one character, some whitespace, a float, perhaps more whitespace, and a newline character. You should read the line character by character, except for the float. In the Java code a BufferedReader is used to do the reading, because it has the method read (which reads a single char) as well as readLine (which reads the remainder of the line, including the newline). In your C code, you need to use a mix of getchar and scanf. In particular, you will need to use getchar to read the operation, then use scanf to read the float, and then use getchar again to read the remaining characters in the line up through the newline.

   In Java code file it is required to catch the IO exceptions. C doesn't, so don't bother.

```
adminuser@adminuser-VirtualBox ~/Desktop/classDemos $ gcc calculator.c  -o calculator
adminuser@adminuser-VirtualBox ~/Desktop/classDemos $ calculator
CSCI 2271 Calculator
+2
*3.5
=
7.000000
/4
=
1.750000
C
=
0.000000
Q
Thank you and goodbye!
adminuser@adminuser-VirtualBox ~/Desktop/classDemos $
```

   When you are done, submit your three C files to Canvas.