CS 61 - Programming Assignment 02

Objective
    To further familiarize you with the basic LC3 instructions;
    to understand the difference between numeric characters and actual numbers;
    to handle two's complement conversions;
    and to perform basic input/output

High Level Description
    Prompt the user to input two single digit numbers; the second will then be subtracted from the first, and the operation reported in the console:
    <first number> - <second number> = <difference>

Low Level Breakdown
    This assignment comprises five tasks:

1. Prompt the user, and read two numeric characters *('0' ... '9')* from the user using Trap x20 (GETC). Echo the characters to the console as they are input, and store them as **character** data in separate registers.
2. Output to the console the operation being performed e.g.
    5 - 7 =
3. Using the same registers, convert the numeric characters into the actual numbers they represent (e.g. convert the ASCII code for '7' into the binary representation of the number 7).
4. Perform the subtraction operation, and determine the sign (+/-) of the result; if negative, determine the magnitude of the result (i.e. take 2's complement)
5. Convert resulting number back to a printable character and print it, together with minus sign if necessary.
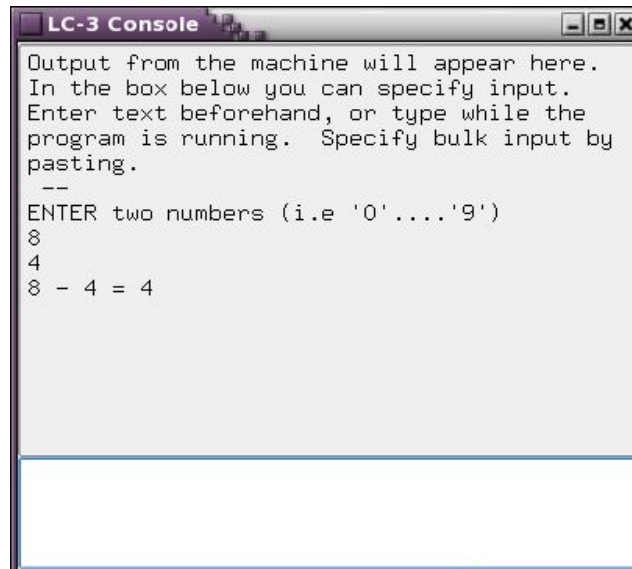
Example with algorithm *(we won't always give you this!)*
1. Program prompts for user input, user enters '5', which is echoed to console and stored in a register.
2. Program prompts for user input, user enters '7', which is echoed to console and stored in a register.
3. Program outputs
    5 - 7 =
4. Program converts '5' into 5 and stores it back in the same register.
5. Program converts '7' into 7 and stores it back in the same register.
6. Program performs (5-7) and stores the result, -2, in a register(obviously as a 2's complement number)
7. Program recognizes that result is negative, converts -2 to 2, and sets flag for minus sign
8. Program converts 2 into '2', and stores it back in same register
9. Program outputs the *two* characters
    -2
    <u>followed by a newline</u>, thus completing the subtraction operation output
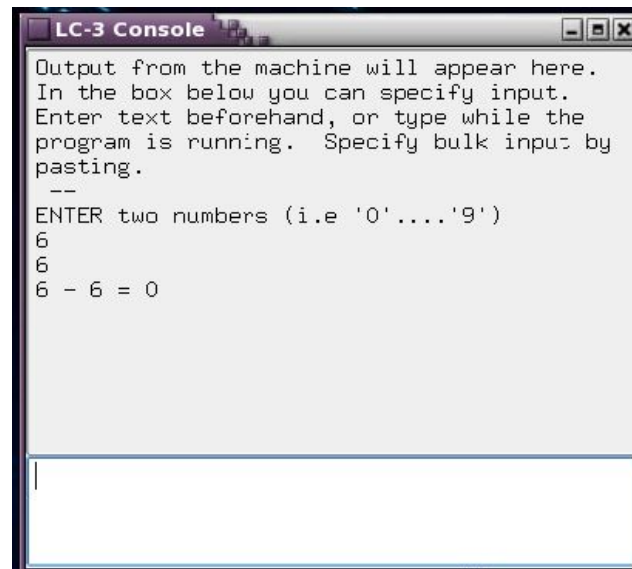
Expected/ Sample output

In this assignment, your output must ***exactly*** match the following, including:

A. the prompt, _followed by newline_
B. each input "echoed" and _followed by a newline_
C. the subtraction operation, including spaces as shown, also followed by a newline:

```
LC-3 Console                        _ ◻ ✕
Output from the machine will appear here.
In the box below you can specify input.
Enter text beforehand, or type while the
program is running.  Specify bulk input by
pasting.
 --
ENTER two numbers (i.e '0'....'9')
8
4
8 - 4 = 4
```
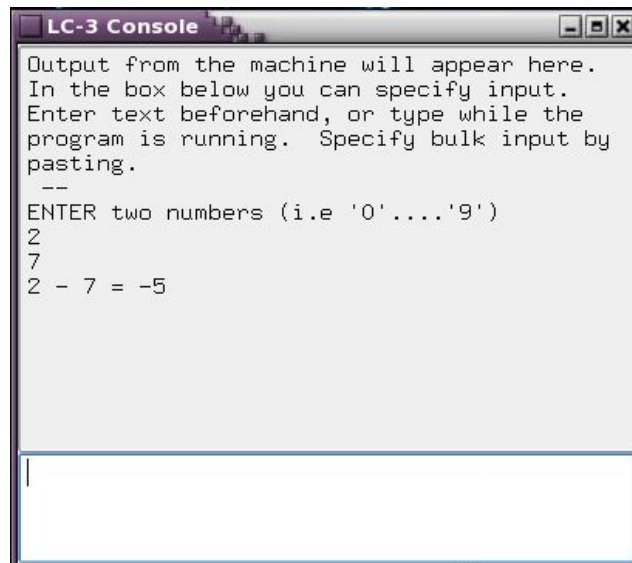
(Difference if Positive)

```
LC-3 Console                        _ ◻ ✕
Output from the machine will appear here.
In the box below you can specify input.
Enter text beforehand, or type while the
program is running.  Specify bulk input by
pasting.
 --
ENTER two numbers (i.e '0'....'9')
6
6
6 - 6 = 0
```

(Difference is Zero)

(Difference is Negative)

Your code will obviously be tested with a range of different operands giving all possible results. Make sure you test your code likewise!

NOTE:
- All console output must be **NEWLINE terminated.**
- We will test only with **positive single digit numeric input**
- **NO** error message is needed for invalid input

Uh...help?
- The instruction "Trap x20" (aka GETC) will *always* store the input character into R0. You cannot specify any other register to receive the keyboard input.
- The instruction "Trap x21" (aka OUT) will *always* print whatever ASCII code is stored in R0. You cannot specify any other register tp output to screen.
- If the user enters '7', the value stored into R0 is the ASCII code b0000 0000 0011 0111
  (= x0037 = '7'), **not** the number 7 = b0000 0000 0000 0111 (= #7).
  Go to www.asciitable.com and see why.
  (conversion between a character and the number it represents will be used repeatedly in this course, so make sure you understand how to do it now!!)
- To take the Two's Complement of a number (i.e. to make a positive number negative or vice versa):
  - Invert the bits *(what assembly instruction does this?)*
  - Add one
- A neat trick in LC3 to copy the value of one register directly to another:
  ADD R5, R6, #0   ; R5 ← (R6) + 0, i.e. R5 ← (R6)
- If the result is negative, remember that you will have to print *two* characters, not one (there is no ASCII code for '-1', right?)
- If you are struggling with writing LC3 code from scratch, try writing the program out in pseudo-code or even C++ first. Then, your only task is to convert the logic/code into LC3.

Submission Instructions

   Submit to GitHub

Comments/Feedback

   Do a "Git pull" to download the results.html file for detailed feedback.

Rubric

- The autograder will attempt to assign partial credit for each test (like those described in the expected output section above), and will report which tests passed and which failed.
  To pass the assignment, you need a cumulative score of >= 8/10.
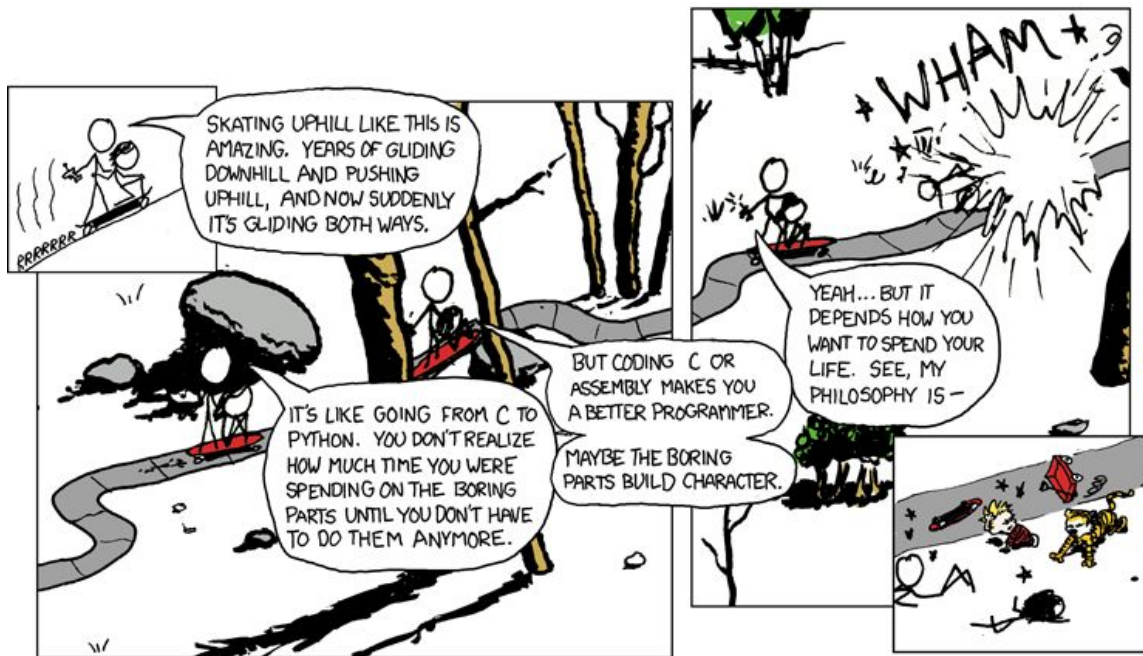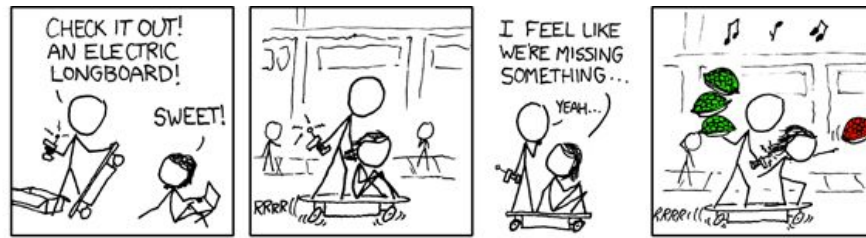 **HOWEVER**: after certain errors in your output (run-time errors, missing newlines, etc), the autograder will be unable to proceed, resulting in a grade of 0/10, with no partial credit.

  This should not be a problem: the problematic output will usually be clearly highlighted in the feedback, so you can fix & resubmit, and hopefully get past the blockage.
  (Unless, of course, you waited until one hour before the deadline to submit, in which case you're stuck with the 0/10 - but you would never do that, would you?)

- **You must use the template we provide** - if you make *any* changes to the provided starter code, the autograder may not be able to interpret the ouput, resulting in a grade of 0.

Comics??! Sweet!!!

Source: http://xkcd.com/409/