CS16, S14, UCSB
Hw2: Worth 50% of Lab02 score (50 total points)

Print this form, staple loose pages together, and write your answers on it.

Accepted: on paper, during *your* lab section, Tuesday July 8th.
Place on the front desk as you walk in, before getting seated.
No email submission allowed.

Name (2 pts): ______________________________________

Umail (2 pts): ______________________@umail.ucsb.edu

Lab Section (2 pts) Circle one: T1 T2 T3

Read to the end of Chapter 2 in the Etter text. Then answer the following questions.

1. First do the "Practice!" exercises on page 38 and check your answers in the
back of the text (this time we look up the location for you - it is page 425).
[If you are using the 3rd edition: exercises page 46; answers page 410.]

Then focus on these statements from problem 2 of those exercises:

```c
int a = 27, b = 6;
float c;
...
c = a / (float)b;
```

a. (6 pts) What is the meaning and what is the effect of the `(float)b` part of the last statement above?

b. (2 pts) What would be the resulting value of c without the `(float)` part (i.e., just c = a / b)?

____________________

c. (6 pts) Would the `(float)` part be necessary if the variables were
declared as follows? Explain your answer.

```c
int b = 6;
float a = 27, c;
```

d. (2 pts) The term used above - "`(float)` part" - is obviously very
non-technical, but there is a specific term to properly call this type of
operator. What is the proper term?

_____________________________
2. Notice on page 41 in the section titled "Overflow and Underflow" the text stresses the fact that actions generated by these conditions are "system dependent."

a. (6 pts) What does "system dependent" mean in terms of these conditions?

b. (6 pts) What can you do to find out how your system responds to an overflow condition?

c. (6 pts) Describe a way you can test how your system responds to an underflow condition.

3. In the section titled "Character I/O" (p. 65) the text notes that scanf and printf can be used with the %c conversion specifier to mimic the behavior of the functions getchar and putchar, respectively. Let c be a variable of type int.

a. (4 pts) Write one proper C statement using scanf instead of getchar that will have exactly the same effect (on the variable c) as the following statement:

   ```c
   c = getchar();
   ```

b. (4 pts) Write one proper C statement using printf instead of putchar that will produce exactly the same output as the following two statements:

   ```c
   putchar(c);
   putchar('\n');
   ```

c. (2 pts) Imagine that the user entered 0 (zero) in part a above, and suppose your printf statement in part b used "%i" instead of "%c" to print this value. What would be the output in this case? [Hint: not 0 - see ASCII codes.]