Read Section 6.1 and 6.2 in your Etter textbook—and review your lecture notes from Thursday Feb 4.

In addition, keep in mind that in the textbook, Etter uses %u as the format specifier for pointers, while in lecture I tend to use %p instead.

The difference is that %u prints the address value as an "unsigned integer", in base 10, while %p prints the address value in hexadecimal.

Then answer these questions:

1. p. 285 shows the program chapter6_1 and p. 285 shows the program chapter6_2

   (In the online version of this homework, you can click on those names as links and get access to the source code in case you want to copy and paste the code and run it.)

   a. (5 pts) p. 284 in your book shows sample output from the program chapter6_1:

   a = 1; address of a = 1245052
   b = 2; address of b = 1245048

   My question to you is this: if you ran this program on CSIL, would the output be the same or different? If different, which parts would be the same, and which parts would be different? And most importantly, explain why.

   (Hint: the answer I'm looking for can be found somewhere in the textbook on pages 284-285.)

   b. (5 pts) Same question, but for the sample output from the program chapter6_2

   a = -858993460; address of a = 1245052
   b = -858993460; address of b = 1245048

   (Hint: again, the answer I'm looking for can be found somewhere in the textbook on pages 284-285.)

Please turn over for more problems
Continued from other side

2. On p. 288, there are four practice problems—the answers to those problems can be found on p. 417 in your textbook.

The following problems should be done in a similar fashion—of course, the answers are NOT in your book—you need to come up with those on your own answers!

Remember that if p is a pointer, its meaning depends on whether it is deferenced with a *.
Also, how you think about it depends on whether it is on the right hand side (rhs) of an assignment statement (rvalue) or the left hand side (lhs) of an assignment statement (lvalue)

- p as an rvalue returns the address stored in p (i.e. the address of what p points to)
- (*p) as an rvalue returns the value of what p points to (e.g. an int, if p is an int *)
- p as an lvalue means store the result of the rhs (which should be an address) in p.
  i.e. make p point to some place new!
- (*p) as an lvalue doesn't change where p points—it changes the value

a. (5 pts)

```c
int a=3, b=4, *ptr;
ptr = &a;
```

b. (5 pts)

```c
int a=5, b=6, *ptr=&a;
b = *ptr;
```

c. (5 pts)

```c
int a=7, b=8, c=9, *ptr=&b;
a = *ptr;
*ptr = c;
```

d. (5 pts)

```c
int a=10, b=11, c=12 *p1=&b, *p2;
p2 = &c;
a = *p1;
p1 = &a;
```

e. (5 pts)

```c
int a=13, b=14, *p1=&a, *p2=&b, *p3;
(*p1) = (*p2);
p3 = p1;
```

f. (5 pts)

```c
int a=15, b=16, *p1=&a, *p2=&b, *p3;
p3 = p1;
p1 = p2;
p2 = p3;
```

End of H11