Day 10 (Mon 02/14)

- exercise 9 review
- day 10 recap questions
- exercise 10

Reminders / announcements:

- HW2 due Friday 2/18
  - No late submissions

- HW 3: soon

- Looking ahead:
  - Midterm project begins Mon 2/28
  - Midterm exam Friday 3/11

Exercise 9 review

Good first step in debugging a program: break main, then run

This gives you control at the very beginning of main

Use next (n) to advance to the next statement

Use step (s) to step into a called function (very important if a function is misbehaving!)

To debug effectively, you need a hypothesis about what is going wrong
- For the transpose function, start with the observation that the print function doesn't print the entire contents of the destination array

Use print (p) to inspect the values of variables, array elements, etc.

Exercise 9 review (continued)

Next issue: the transpose function doesn't seem to correctly transpose the elements of the original array

Step into the call to transpose

Inspect "shape" and contents of the two arrays

print start[0]
print start[1]
print start[2]

Look carefully at the code at line 13 (do the array subscripts make sense?)

Exercise 9 review (continued)

Debuggers are not magic

They will not tell you what's wrong with your code
  - because they have no idea what your code is supposed to do!

They are VERY useful for seeing what your code is actually doing
  - they make the internal state of the program visible

Pro tip: learn how to set breakpoints

  - break <name of function>
  - break <name of source file>:<line number>

Use the "continue" command (c) to run the program until the next breakpoint is reached

Day 10 recap questions

1. What is a pointer?

2. If a is an int variable, and p is a variable whose type is pointer-to-int, how do you make p point to a?

3. If p is a pointer-to-int variable that points to an int variable a, how can you access the value of a or assign a value to a without directly referring to a? Show examples of printing the value of a and modifying the value of a, but without directly referring to a.

4. When calling scanf, why do you need to put a & symbol in front of a variable in which you want scanf to store an input value?

5. Trace the little program below and determine what the output will be.

1. What is a pointer?

It is a value that represents the *location* ("address") of another variable.
So, it is a way of referring *indirectly* to another variable.

Many uses! One general application of pointers is to allow one function to refer to a variable defined in a different function. (Ordinarily, one function cannot refer to another function's variables.)

Much danger! If a pointer refers to a variable that no longer exists, it could allow the program to access invalid memory.

2, 3.

```
int a;
int *p;

p = &a;                 // p "points to" a
*p = 42;                // store 42 in the variable p points to, which is a
printf("%d\n", *p);  // will print 42
printf("%d\n", a);   // will print 42
```

4.

We use &, the address-of operator, so that we can pass a pointer
to a variable to scanf.

This allows scanf to access the variable *indirectly*, since it has no way of directly referring to the variable by name.

5.

```
int func(float ra[], float x, float *y) {
    ra[0] += 10;
    x *= 20;
    *y += 30;
    return 40;
}
int main() {
    float a = 1;
    float b = 2;
    float c[] = {3, 4, 5, 6};
    int d;
    d = func(c, a, &b);
    printf("%.2f, %.2f, %.2f, %d\n", a, b, c[0], d);
}
```