Computer Programming: Skills and Concepts
Solutions for Tutorial 3 (week 5)

Descartes

The students have been using the `descartes` graphics library in their current lab. Many of them have been struggling with the difference between how a function prototype is *declared*, and how it is *used* ... hence many sections of code written in the Labs contain sequences of re-declarations, and statements that look like mixed declarations/calls.

Somehow the fact that they are working with structured types has reinforced their confusion between the declaration of a function, and the use of that function (this confusion is seen every year). If you can spend some time discussing the difference with them (with reference to the `descartes.h` file available on the course website) that would be a great start.

This question is to give them a bit more practice with using the Descartes functions. You should make sure to do this question in two steps, first (i) using the variables `q, r, seg` to hold intermediate objects such as the point `(40, 300)` before `seg` is drawn on the screen; then (ii) showing how we can draw the segment with a single statement and using no variables.

Solutions are in `tutw5ai.c` and `tutw5aii.c`.

Very important, when finished with (ii), is to mention that the reason we could get away with no variables at all is because the particular segment is fixed from the start. We would need at least two `point_t` variables (though not necessarily a `lineSeg_t` variable) if we were defining a segment depending on user-input (that's actually a task they have been doing in the current lab).

Descartes 2

This question is quite a tough one. It is reasonable to expect they can code up `IsRectangleOK` (assuming they have surmounted their difficulties in separating use-of-a-function from function-prototype). The coding of `Rectangle2` will be more difficult as they need to do a reasonable amount of logical reasoning as well as deal with the use of the Descartes functions. So possibly a two-stage development of `Rectangle2` is a good idea: first developing the logical structure of the code; then dealing with implementation using `XCoord`, `YCoord`, `Point` and the original `Rectangle` function.

Both functions are coded up in `tutw5b.c`. This file also contains a development of 3 Examples (including the two suggested to the students) in its `main`.

The file also uses `FillRectangle()` to show the rectangles as produced by `Rectangle()` and `Rectangle2()`.

Pointers

Consider the following code:

```
int a = 10;
int b = 5;
int *p = &a;
int *q = &b;
int c = *p;
c = c + 1;
*p = *p + 2;
int d = *p;
*p = *q;
int e = *p;
b = b + 5;
int f = b;
int g = *p;
int h = *q;
```

What is the value of b, c, d, e, f, g, and h?

answer: `10, 11, 12, 5, 10, 5, 10`

Tutors, please draw pictures with boxes for the various variables (as in lecture slides 3 and 8), and track the changes as the various statements are executed.

The goal of this question is to get students to understand pointers (like `p`) and their values (like `*p`).