EXERCISES ON STRUCTURES

1. What is wrong with the following C declarations?

   A. `struct point ( double x, y )`
   B. `struct point { double x, double y };`
   C. `struct point { double x; double y }`
   D. `struct point { double x; double y; };`
   E. `struct point { double x; double y; }`

2. What is wrong with the following C declarations?

   A. `typedef struct { double x; double y } Point;`
   B. `typedef { double x; double y; } Point;`
   C. `typedef struct { double x; double y; };`
   D. `typedef struct { double x; double y; } Point;`

3. What is the difference among the following three programs?

   (a)
   ```
   #include <stdio.h>
   struct point { double x; double y; };
   int main(void) {
      struct point test;
      test.x = .25; test.y = .75;
      printf("[%f %f]\n", test.x, test.y);
      return 0;
   }
   ```

   (b)
   ```
   #include <stdio.h>
   typedef struct { double x; double y; } Point;
   int main(void) {
      Point test;
      test.x = .25; test.y = .75;
      printf("[%f %f]\n", test.x, test.y);
      return 0;
   }
   ```

   (c)
   ```
   #include <stdio.h>
   typedef struct { double x; double y; } Point;
   int main(void) {
      Point test = {.25, .75};
      printf("[%f %f]\n", test.x, test.y);
      return 0;
   }
   ```

4. Provide an implementation of a function POINTshow() so that the following program is functionally equivalent to the three programs above.

   ```
   #include <stdio.h>
   typedef struct { double x; double y; } Point;
   int main(void) {
     Point test = {.25, .75};
     POINTshow(test);
     return 0;
   }
   ```

5. Provide an implementation of a function POINTdist() that computes the Euclidean distance between two Points.

6. Provide an implementation of a function POINTeq() that returns 1 if the two points are "equal"; and 0 otherwise. With floating point values it doesn't make much sense to test for exact equality; instead check to see if the distance between the points is less than 0.000001.

7. Define a type Rect for rectangles that are parallel to the axes in a Cartesian coordinate system. Represent a rectangle by its lower left and upper right endpoints using the Point type above.

8. Write a function RECTarea() that computes the area of a rectangle.

9. Write a function that returns 1 if a point falls within a rectangle, 0 otherwise. Use the Point and Rectangle types above.

10. Write a function that returns 1 if the first rectangle is completely contained inside the second rectangle, and 0 otherwise. Hint: check if the lower left and upper right endpoints of the first rectangle fall within the second rectangle.

11. Write a program that reads in a list of points (given by their x and y coordinates) and determines the pair that is the farthest apart. Hint: store them in an array and use the POINTdist() function.