CIS 330: Project #2C
Assigned: April 23rd, 2018
Due April 28th, 2018
(which means submitted by 6am on April 29th, 2018)
Worth 4% of your grade

Assignment: You will implement 3 structs and 9 functions. All the files (.c, .h, Makefile, correct output, grader program) needed for this project are available on the course website.

The three structs are Rectangle, Circle, and Triangle, and are described below.

The 3 structs refer to 3 different shapes: Triangle, Circle, and Rectangle.
For each shape, there are 3 functions: Initialize, GetArea, and GetBoundingBox.
You must implement 9 functions total (3*3).

The prototypes for these 9 functions are available in the file prototypes.h

There is also a driver program (driver_2C.c) that calls your functions and prints the results to stdout. The correct ground-truth output for the driver program is contained in the driver_output file.

Again, your job is to define 3 structs and 9 functions. The comments below clarify the format of the Rectangle, Circle, and Triangle, as well as the convention for GetBoundingBox, and an example of accessing data members for pointers to structs.

== Rectangle ==

The rectangle has corners (minX, minY), (maxX, minY), (minX, maxY), (maxX, maxY).
Its area is (maxX-minX)*(maxY-minY).
Its bounding box is from minX to maxX in X, and minY to maxY in Y.

== Circle ==

The circle has an origin (x and y) and a radius.

Its area is 3.14159*radius*radius.
Its bounding box is from (x-radius) to (x+radius) in X, and (y-radius) to (y+radius) in Y.

== Triangle ==

The triangle always has two points at the minimum Y-value. The third point's Y-value is at the maximum Y-value, and its X-value is at the average of the X's of the other two points. Saying it another way, the first two points form the "base", and the third point is "height" above it.

Thus, the area of the triangle is (pt2X-pt1X)*(maxY-minY)/2;
And the bounding box is from pt1X to pt2X in X, and from minY to maxY in Y.

== GetBoundingBox ==

The GetBoundingBox functions take a double * as an argument. If a shape has its minimum X at "a", its maximum X at "b", its minimum Y at "c", and its maximum Y at "d", then it should do something like:

```
void GetCircleBoundingBox(Circle *, double *bbox)
{
   bbox[0] = a;
   bbox[1] = b;
   bbox[2] = c;
   bbox[3] = d;
}
```

== Working with pointers to structs ==

We reviewed the way to access struct data members in class, which was with the "." operator. We did not review the way to access struct data members when you have a *pointer* to a struct. And the 9 function prototypes all use pointers to structs. It is done with the ->.

So:
```
typedef struct
{
   int X;
} Y;
int main()
{
  Y y;
  Y *y2;
  y2 = &y;
  y.x = 0;
  y2->x = 1;
}
```

== What to modify ==

You will need to modify my_struct.h and my_struct.c. You should **not** modify prototypes.h or driver_2C.c. If you modify they latter two files, you will have points deducted. The grader program (grader.sh) will reveal whether these files have been modified.

== Success ==

You should compile your program using the provided Makefile. The executable will be named “project_2C”.

Then run your program as:

```
./project_2C > my_output
```

and call:

```
diff my_output driver_output
```

If diff returns no differences, then your program produces the correct output.

== What to turn in ==

Before you submit, make sure to test your code on ix-dev. You should execute the provided grader program script (grader.sh) prior to submitting. It should be called within your project directory on ix-dev as follows:

```
 ./grader.sh
```

If you pass all tests, that only assures that your code compiles properly with the correct input and output, and follows the requirements of this prompt. However, your actual source code (my_struct.h and my_struct.c) will still be graded for good programming practices. The project will be graded on ix-dev.

Then, make a file called “README”.
In that file, notify the reader whether you think your program is correct or not.

Please submit a tarball named 2C_turnin.tar with the following files:

```
% tar -cvf 2C_turnin.tar my_struct.h my_struct.c my_output README
```