CMPS109 Spring 2018 : Lab 2

In this lab you will implement a two and three-dimension geometric bounds checker in C.

This lab is worth 7% of your final grade.

Submissions are due NO LATER than 23:59, Sunday April 15, 2018.

Late submissions will not be graded.

Background Information

The web is awash with information on the mathematics required to do the simple (and some not-so simple) calculations required to complete this lab.

Teaching Servers & Accessing Your Home Directory

See Lab 1 for details: https://classes.soe.ucsc.edu/cmps109/Spring18/SECURE/CMPS109-Lab1.pdf

Setup

SSH in to any of the CMPS109 teaching servers using your CruzID Blue credentials:

```
$ ssh <cruzid>@<server>.soe.ucsc.edu
```

(use Putty http://www.putty.org/ if on Windows)

Authenticate with Kerberos:

```
$ kinit <cruzid>@CATS.UCSC.EDU
```

Authenticate with AFS:

```
$ aklog
```

Create a suitable place to work: ***(only do this the first time you log in)***

```
$ mkdir -p ~/CMPS109/Lab2
$ cd ~/CMPS109/Lab2
```

Install the lab environment: ***(only do this once)***

```
$ tar xvf /var/classes/CMPS109/Spring18/Lab2.tar.gz
```

Make the skeleton system:

```
$ make grade
```

What to submit

In a command prompt:

```
$ cd ~/CMPS109/Lab2
$ make submit
```

This creates a gzipped tar archive named `CMPS109-Lab2.tar.gz` in your home directory.

UPLOAD THIS FILE TO THE APPROPRIATE CANVAS ASSIGNMENT.

Requirements

Consider a space (the “arena”) in which one or more shapes reside. When a shape moves, you are required to calculate whether or not the shape remains wholly contained within the bounds of the arena.

You are supplied with a number of sample test scripts that provide information from which a test harness (for which you do not have the source code) creates an arena and one or more shapes within it.

The test scripts have the following format, where any line starting with the “#” character is regarded as a comment, and blank lines are ignored:

```
# The arena, a rectangle
arena Polygon 8,8,0 8,-8,0 -10,-10,0 -10,10,0

# A shape within the arena, a 4x4 square
shape Polygon 4,4,0 4,-4,0 -4,-4,0 -4,4,0

# Another shape within the arena, a circle of radius 4
shape Circle 0,0,0 4

# Move shape zero (the square) to coordinates 1,1,0 leaving it inside the arena
0 1,1,0 true

# Move shape one (the circle) to coordinates 5,5,0 leaving at least some part
# of it outside the arena
1 5,5,0 false
```

Note that all vertices and points are in the three-dimensional coordinate system, the z coordinate being zero when working in two-dimensions.

Also note that syntax errors cause the test harness to silently exit without warning.

What you need to do

You will modify `bounds.c` to fully implement the `setup()` and `move()` methods.

More importantly, you will modify the simple scripts supplied to include far more rigorous tests of your implementation. You can also create your own test scripts; these will be included in your submission but will not be executed by the automated grading system and not count towards your grade for this lab.

To execute a test stand-alone after running `make`:

```
$ ./bounds <name>.test
```

You can execute the tests in any sequence you like, but it is recommended you undertake them in this order:

```
trivial.test
single1.test
single2.test
double1.test
double2.test
tripple.test
pentagon1.test
pentagon2.test
pentagon3.test
reuleaux1.test
reuleaux2.test
threedim1.test
threedim2.test
```

Note that the last two tests have three-dimensional arenas and shapes, all the others are two-dimensional.

Also note that if you wish to use the standard C mathematics library in your solution, you will need to modify the `LIBS` variable in `Makefile.libs`.

Grading Scheme

The following aspects will be assessed by executing your code on a machine with an identical configuration to the CMPS109 teaching servers:

1. (70 Marks) **Does it work?**

    a. Geometric Calculations (60 marks)
    b. Code free of warnings (10 marks)

For a, marks are deducted for any geometric calculations failing to produce the correct answer.

Note that the test scripts used by the automated grading system will **NOT** be the same as the trivial skeleton scripts provided in the lab environment you installed.

2. (-100%) **Did you give credit where credit is due?**

    a. Your submission is found to contain code segments copied from on-line resources and you failed to give clear and unambiguous credit to the original author(s) in your source code (-100%)

    b. Your submission is determined to be a copy of another CMPS109 student's submission (-100%)

    c. Your submission is found to contain code segments copied from on-line resources that you did give a clear an unambiguous credit to in your source code, but the copied code constitutes too significant a percentage of your submission:

        - < 25% copied code — No deduction
        - 25% to 50% copied code — (-50%)
        - 50% to 75% copied code — (-75%)
        - > 75% — (-100%)

§

