Solid Geometry (Due 18 Sep 2020)

This assignment is on object oriented programming. You will be developing several classes that are fundamental in Solid Geometry - Point, Sphere, Cube, and Cylinder. In main() you will test the various functions that you have written for the classes.

The file (Geometry.py) that you will be submitting will have the following structure. You may **NOT** change the names of the functions but you may add as many helper functions as needed. You will follow the standard coding conventions in Python.

```
import math

class Point (object):
  # constructor with default values
  def __init__ (self, x = 0, y = 0, z = 0):

  # create a string representation of a Point
  # returns a string of the form (x, y, z)
  def __str__ (self):

  # get distance to another Point object
  # other is a Point object
  # returns the distance as a floating point number
  def distance (self, other):

  # test for equality between two points
  # other is a Point object
  # returns a Boolean
  def __eq__ (self, other):

class Sphere (object):
  # constructor with default values
  def __init__ (self, x = 0, y = 0, z = 0, radius = 1):

  # returns string representation of a Sphere of the form:
  # Center: (x, y, z), Radius: value
  def __str__ (self):

  # compute surface area of Sphere
  # returns a floating point number
  def area (self):

  # compute volume of a Sphere
  # returns a floating point number
  def volume (self):

  # determines if a Point is strictly inside the Sphere
  # p is Point object
  # returns a Boolean
  def is_inside_point (self, p):

  # determine if another Sphere is strictly inside this Sphere
  # other is a Sphere object
  # returns a Boolean
  def is_inside_sphere (self, other):

  # determine if a Cube is strictly inside this Sphere
  # determine if the eight corners of the Cube are strictly
  # inside the Sphere
  # a_cube is a Cube object
  # returns a Boolean
  def is_inside_cube (self, a_cube):

  # determine if a Cylinder is strictly inside this Sphere
  # a_cyl is a Cylinder object
  # returns a Boolean
  def is_inside_cyl (self, a_cyl):
```

```
    # determine if another Sphere intersects this Sphere
    # other is a Sphere object
    # two spheres intersect if they are not strictly inside
    # or not strictly outside each other
    # returns a Boolean
    def does_intersect_sphere (self, other):

    # determine if a Cube intersects this Sphere
    # the Cube and Sphere intersect if they are not
    # strictly inside or not strictly outside the other
    # a_cube is a Cube object
    # returns a Boolean
    def does_intersect_cube (self, a_cube):

    # return the largest Cube object that is circumscribed
    # by this Sphere
    # all eight corners of the Cube are on the Sphere
    # returns a Cube object
    def circumscribe_cube (self):

  class Cube (object):
    # Cube is defined by its center (which is a Point object)
    # and side. The faces of the Cube are parallel to x-y, y-z,
    # and x-z planes.
    def __init__ (self, x = 0, y = 0, z = 0, side = 1):

    # string representation of a Cube of the form:
    # Center: (x, y, z), Side: value
    def __str__ (self):

    # compute the total surface area of Cube (all 6 sides)
    # returns a floating point number
    def area (self):

    # compute volume of a Cube
    # returns a floating point number
    def volume (self):

    # determines if a Point is strictly inside this Cube
    # p is a point object
    # returns a Boolean
    def is_inside_point (self, p):

    # determine if a Sphere is strictly inside this Cube
    # a_sphere is a Sphere object
    # returns a Boolean
    def is_inside_sphere (self, a_sphere):

    # determine if another Cube is strictly inside this Cube
    # other is a Cube object
    # returns a Boolean
    def is_inside_cube (self, other):

    # determine if a Cylinder is strictly inside this Cube
    # a_cyl is a Cylinder object
    # returns a Boolean
    def is_inside_cylinder (self, a_cyl):

    # determine if another Cube intersects this Cube
    # two Cube objects intersect if they are not strictly
    # inside and not strictly outside each other
    # other is a Cube object
    # returns a Boolean
    def does_intersect_cube (self, other):

    # determine the volume of intersection if this Cube
    # intersects with another Cube
    # other is a Cube object
    # returns a floating point number
    def intersection_volume (self, other):
```

```
    # return the largest Sphere object that is inscribed
    # by this Cube
    # Sphere object is inside the Cube and the faces of the
    # Cube are tangential planes of the Sphere
    # returns a Sphere object
    def inscribe_sphere (self):

  class Cylinder (object):
    # Cylinder is defined by its center (which is a Point object),
    # radius and height. The main axis of the Cylinder is along the
    # z-axis and height is measured along this axis
    def __init__ (self, x = 0, y = 0, z = 0, radius = 1, height = 1):

    # returns a string representation of a Cylinder of the form:
    # Center: (x, y, z), Radius: value, Height: value
    def __str__ (self):

    # compute surface area of Cylinder
    # returns a floating point number
    def area (self):

    # compute volume of a Cylinder
    # returns a floating point number
    def volume (self):

    # determine if a Point is strictly inside this Cylinder
    # p is a Point object
    # returns a Boolean
    def is_inside_point (self, p):

    # determine if a Sphere is strictly inside this Cylinder
    # a_sphere is a Sphere object
    # returns a Boolean
    def is_inside_sphere (self, a_sphere):

    # determine if a Cube is strictly inside this Cylinder
    # determine if all eight corners of the Cube are inside
    # the Cylinder
    # a_cube is a Cube object
    # returns a Boolean
    def is_inside_cube (self, a_cube):

    # determine if another Cylinder is strictly inside this Cylinder
    # other is Cylinder object
    # returns a Boolean
    def is_inside_cylinder (self, other):

    # determine if another Cylinder intersects this Cylinder
    # two Cylinder object intersect if they are not strictly
    # inside and not strictly outside each other
    # other is a Cylinder object
    # returns a Boolean
    def does_intersect_cylinder (self, other):

  def main():
    # read data from standard input

    # read the coordinates of the first Point p

    # create a Point object

    # read the coordinates of the second Point q

    # create a Point object

    # read the coordinates of the center and radius of sphereA

    # create a Sphere object

    # read the coordinates of the center and radius of sphereB
```

```
    # create a Sphere object

    # read the coordinates of the center and side of cubeA

    # create a Cube object

    # read the coordinates of the center and side of cubeB

    # create a Cube object

    # read the coordinates of the center, radius and height of cylA

    # create a Cylinder object

    # read the coordinates of the center, radius and height of cylB

    # create a Cylinder object


    # print if the distance of p from the origin is greater
    # than the distance of q from the origin


    # print if Point p is inside sphereA

    # print if sphereB is inside sphereA

    # print if cubeA is inside sphereA

    # print if cylA is inside sphereA

    # print if sphereA intersects with sphereB

    # print if cubeB intersects with sphereB

    # print if the volume of the largest Cube that is circumscribed
    # by sphereA is greater than the volume of cylA


    # print if Point p is inside cubeA

    # print if sphereA is inside cubeA

    # print if cubeB is inside cubeA

    # print if cylA is inside cubeA

    # print if cubeA intersects with cubeB

    # print if the intersection volume of cubeA and cubeB
    # is greater than the volume of sphereA

    # print if the surface area of the largest Sphere object inscribed
    # by cubeA is greater than the surface area of cylA


    # print if Point p is inside cylA

    # print if sphereA is inside cylA

    # print if cubeA is inside cylA

    # print if cylB is inside cylA

    # print if cylB intersects with cylA

  if __name__ == "__main__":
    main()
```

Note that in this program you will be checking for the equality of two floating point numbers. Since there is a finite precision in the representation of floating point numbers, it is not always possible to determine exact equality. A working solution is to determine equality is to take the difference of the floating point numbers and see if the difference is less than a pre-determined tolerance. This tolerance is arbitrary and is often dictated by the problem that you are trying to solve. Here is a function that tests for equality of two floating point numbers.

```
def is_equal (a, b):
  tol = 1.0e-6
  return (abs (x - y) < tol)
```

Input: You will be reading your input from standard input in the following format geometry.in.

```
-3.0 2.0 1.0                    # coordinates of p
2.0 -1.0 3.0                    # coordinates of q
2.0 1.0 3.0 4.0                 # center and radius of sphereA
-1.0 -2.0 -3.0 5.0              # center and radius of sphereB
2.0 1.0 -3.0 4.0                # center and side of cubeA
3.0 2.0 -4.0 3.0                # center and side of cubeB
-2.0 1.0 -3.0 5.0 4.0           # center, radius, and height of cylA
1.0 5.0 3.0 4.0 2.0             # center, radius, and height of cylB
```

Output: You will print your output to standard out in the following format geometry.out:

```
Distance of Point p from the origin (is / is not) greater than the distance of Point q from the origin

Point p (is / is not) inside sphereA
sphereB (is / is not) inside sphereA
cubeA (is / is not) inside sphereA
cylA (is / is not) inside sphereA
sphereA (does / does not) intersect sphereB
cubeB (does / does not) intersect sphereB
Volume of the largest Cube that is circumscribed by sphereA (is / is not) greater than the volume of cylA

Point p (is / is not ) inside cubeA
sphereA (is / is not) inside cubeA
cubeB (is / is not) inside cubeA
cylA (is / is not) inside cubeA
cubeA (does / does not) intersect cubeB
Intersection volume of cubeA and cubeB (is / is not) greater than the volume of sphereA
Surface area of the largest Sphere object inscribed by cubeA (is / is not) greater than the surface area of cylA

Point p (is / is not) inside cylA
sphereA (is / is not) inside cylA
cubeA (is / is not) inside cylA
cylB (is / is not) inside cylA
cylB (does / does not) intersect cylA
```

For this assignment you may work with a partner. Both of you must read the paper on Pair Programming and abide by the ground rules as stated in that paper. If you are working with a partner then only one of you will submit the code. Make sure that in the header in HackerRank that you have your name and UT EID and your partner's name and UT EID. If you are working alone then you will just have your name and your UT EID.

Use the *HackerRank* platform to submit your code. We should receive your work by 11 PM on Friday, 18 Sep 2020. There will be substantial penalties if you do not adhere to the guidelines. HackerRank will not assign late penalties (if any), we will make those adjustments.

- Your code must run before submission.
- You should be submitting your file through the web based program - HackerRank. We will not accept files e-mailed to us.