Major Assignment 5

ENGINEER 1D04
Dr. William Farmer and Dr. Spencer Smith
McMaster University, Winter 2012

Revised: June 19, 2013

Please use AutoMarker (`automarker.mcmaster.ca`) and Avenue to acquire, test, package and submit your assignment. The procedure for submitting assignments is summarized on Avenue, with additional details provided by AutoMarker. **Please frequently back up your work by creating a submission package in AutoMarker**. This will provide a chance to recover your work in the event of an equipment failure.

Background

In geometry, a 2-dimensional convex polygon is any closed figure with $n > 2$ vertices (and $n > 2$ edges), in which every internal angle is less than 180 degrees and no two edges cross each other. For example, any parallelogram is a valid convex polygon with $n = 4$. We can define such a polygon in the 2-D cartesian plane by giving a list of vertex points (x, y pairs), where it is assumed that the list of vertices travels "clockwise" around the polygon, as follows:

$$v = p_0, p_1, ..., p_{n-1} = (x_0, y_0), (x_1, y_1), ..., (x_{n-1}, y_{n-1})$$

The polygon edges are then defined to be the lines between each p_i, p_{i+1} pair. By definition, $x_n = x_0$ and $y_n = y_0$, which "closes" the polygon with the line p_{n-1}, p_0. In Python, we can use this definition to specify a parallelogram (for example) with the following list of vertices:

```
v = [ [0.0, 0.0], [0.5, 1.5], [2.5, 1.5], [2.0, 0.0] ]
```

A polygon defined this way can be easily rotated in place by some angle θ using a "rotation matrix." In 2-D space, the operation is relatively simple and reduces to the following two operations for each point p in v:

$$x_{new} = x \cdot \cos(\theta) - y \cdot \sin(\theta),\ y_{new} = x \cdot \sin(\theta) + y \cdot \cos(\theta)$$

A polygon can also be shifted by a in the x- direction and b in the y- direction by applying the following operations for each point p in v:

$$x_{new} = x + a,\ y_{new} = y + b$$

Design, implement, and test a program that satisfies the requirements below.

****IMPORTANT!!!**:** This assignment will be run through an automated testing program to be graded. Function syntax in your program must be **exactly** as specified, including spelling, capitalization, and the order of function parameters. **DO NOT** include a `main` function. Failure to precisely follow the requirements below will result in a **significant loss of marks**.

Requirements

1. The library includes a Python class named `ConvexPolygon`.

2. The `__init__` method for the class takes the first formal parameter `self` and a list of `[x, y]` pairs v (as in the parallelogram example above), and stores the list as a class instance variable. Note that the length of the list is not fixed!

3. The class contains accessors for the vertices v and the number of vertices n (i.e. the length of v). The names of the accessors are `getVertices` and `getN`, respectively.

4. The class contains a mutator `setVertex(self, i, x, y)` that takes the first formal parameter `self`, an int `i`, and floats `x` and `y` and updates the `i`-th vertex in v to be`[x, y]` (ie. update p_i in v). Note that the length of v does not change!

5. The class contains a mutator `rotate(self, theta)` that takes the formal parameter `self` and a float `theta` in **radians** and rotates each vertex in v using the rotation operation given above.

6. The class contains a method `shift(self, x, y)` that takes the first formal parameter `self` and floats `x` and `y` and returns a new `ConvexPolygon` object that represents the shifting of the `ConvexPolygon` object represented by self in the x and y direction by `x` and `y` units respectively. The shift operation is given above.

7. The program requires very little besides the function definitions. There is no `main()`.

8. The program does not read anything from standard input or write anything to standard output. That is, the program does not interact with the user who invokes it.

9. The program is written in Python in a module, NOT in the Python Shell. To create a new module in IDLE, go to File $\longrightarrow$ New Window. You must save this file with a .py extension. For more information on submitting your program, click the "AutoMarker Instructions" button above.

10. Your name, MacID, student number, and the date are given in comments at the top of your Python (.py) file before your program.

11. Your answers to the design questions and the test plan (see below) are given in comments at the bottom of your Python (.py) file after your program.

12. Your program MUST have valid Python syntax and it must run without errors. Ensure that your program runs properly by running it before you submit.

13. You must sign out with a TA or IAI after you have submitted your lab at the submission station. Failure to do so could result in a zero.

Design and Implementation Instructions

1. Use the `math` library to implement the sin and cos functions.
2. You may assume that inputs will be within a valid range and of the correct type, and that exception handling is not necessary.
3. Follow the function syntax **EXACTLY** as given, including spelling, capitalization and the order of function parameters.

Design Question

The calculation of x_{new} depends on the original value of y, and the calculation of y_{new} depends on the original value of x. How will your program ensure that the correct calculation is performed for both values?

Test Plan

Produce a test plan with test cases for each of the methods/functions defined above using the following form:

```
Test:  i for function j
Input:  inputs for function j
Expected Output:  expected output for function j
```

For mutators, Expected Output will be the expected values for the instance variables that the mutator is changing. You should have enough test cases to adequately support the argument that your code is correct. Your test cases should cover as many different classes of input cases as possible, including boundary cases. Your test plan should includes case(s) where your expected output is a failure, as appropriate.