CMPT 103 – Lab #8

General Information

Python version and IDE:	Python 3 / Wing IDE 101
Allocated lab time:	5 hours and 40 minutes
Due date:	As stated on Blackboard
Lab weight:	5%

Extensions:
All students are expected to complete the lab assignment by the end of their second lab period. Any submissions after this deadline will be considered late and will receive a 25% late penalty UNLESS students have attended the lab and have received an extension from their lab instructor in which case no penalty will be given. NO assignments will be accepted 24 hours after the lab has ended.

Topics

- ✓ Object-oriented programming
- ✓ Building modules

Submission

- ✓ **All the code files (.py) should be submitted electronically** to the class Blackboard site.
- ✓ Good programming style in your programs is important.
- ✓ **Comment your code** or you will lose marks. Comments are **required** for:
  - EACH program indicating your name and program name.
  - EACH function indicating function purpose, syntax (usage), parameters, and return values.
  - EACH variable that does not have a descriptive name.
  - Any code for which the purpose may be unclear.

Assignment

This two-week assignment has been designed to give you practice with classes and modules. In your solution to this lab, name all instance variables and methods as described.

1. [10 marks] In a new file named `geometry.py`, create two classes: `Point` and `Rectangle`.
   - `Point`: a class consisting of instance variables `x` and `y` (integers)
   - `Rectangle`: a class consisting of instance variables (a) `origin` (**a `Point`**) for the lower-left corner, (b) `width` (an integer), and (c) `height` (an integer)

2. [20 marks] Implement each method described below for the appropriate class. The prototypes and behavior must match the descriptions.
   - `__init__(self)`: implement this method **for both classes**. For `Point`, initialize your instance variables to the default values of (0, 0); for `Rectangle`, use the default values of (0, 0), 1, and 1 for origin, width, and height, respectively.
   - `__repr__(self)`: implement this method **for both classes** to return a string representation of the object (see the sample output on the last page).
   - `update(self, x, y)`: implement this method **for the `Point` class** to update the x and y coordinates to `x` and `y`, respectively.
   - `update(self, origin, width, height)`: implement this method **for the `Rectangle` class** to update the values of the origin, width, and height to the new values given by the parameters.
3. [30 marks] Implement each method described below for the `Rectangle` class. The prototypes and behavior must match the descriptions.
   - `area(self)`: `Rectangle` method: returns the (numeric) area of the rectangle.
   - `is_square(self)`: `Rectangle` method: returns `True` if the rectangle is a square and `False` otherwise.
   - `get_coords(self)`: `Rectangle` method: returns a list with the corner coordinates of the rectangle in the following order: lower left, upper left, upper right, and lower right. The list must contain **four `Point` objects**.
   - `get_visual(self)`: `Rectangle` method: returns a **string representation** of the rectangle as an unfilled rectangle. Use '–' for the top and bottom lines, '|' for the sides, and '+' for the corners. See the sample output (last page) for an example.
4. [20 marks] Write a function `coords_to_rectangle(rect, coords)`: Given a valid value for `coords` (described later), this function updates the origin, width, and height of the `rect` parameter (a `Rectangle`) based on the coordinates (`Point`s) given in the list. The order of coordinates in the list will be the same as with `get_coords`: lower left, upper left, upper right, and lower right. It returns `False` if the coordinates do not describe a rectangle and `True` if they do. If the coordinates do not describe a rectangle, it leaves `rect` unchanged.
5. [20 marks] In a separate file, `L8demo.py`, write a test program to verify that your functions perform as expected. You have already used a Python file, `graphics.py`, as a module. In `L8demo.py`, you'll use `geometry.py` as a module by writing

   ```
   from geometry import *
   ```

   Save this file (`L8demo.py`) in the same directory as `geometry.py`.

   The demo program must:

   a. Print a title to the screen.
   b. Test the `__init__` methods by creating and displaying new `Point` and `Rectangle` objects.
   c. Use `Point`'s `update` method to update the position to (2, 3).
   d. Display the updated `Point`.
   e. Use `Rectangle`'s `update` method to update the origin, width, and height to (1, 1), 5, and 8.
   f. Display the updated `Rectangle`.
   g. Create a new `Rectangle` with a (different) position and dimensions (of your choice).
   h. Test the `area` function on the new rectangle and print the result to the screen.
   i. Test `get_coords` on your rectangle, and print out the result to the screen.
   j. Create 2 coordinate lists, one that does not represent a rectangle and one that represents one. Test the `coords_to_rectangle` function and print the results to the screen.
   k. Test the `is_square` function on a rectangle that is a square and one that is not. Print the results to the screen.
   l. Test `get_visual` on two of your rectangles and print the results to the screen.
   m. Test some other possible sources of errors.

See the sample output below as a guide.

All lines of output produced by the test program's `main` are indicated with a "`->`".

All other output is generated by printing the return values of the functions that you were required to write.

```
-> Testing Program For Geometry Module
-> Point Default
0 0
-> Rectangle Default
0 0
1
1
-> Point
2 3
-> Rectangle
1 1
5
8
-> New rectangle: r1
3 2
7
3
-> The area is of r1 is 21
-> checking get_coords on r1...
3 2
3 5
10 5
10 2

-> Testing coords_to_rectangle....
-> The following coords are not a rectangle.
-> 1 1
-> 2 1
-> 3 1
-> 4 1
-> The result of the function on the list above was:
False

-> The following coords give the following rectangle.
-> 1 1
-> 1 4
-> 6 4
-> 6 1

-> The rectangle it describes is:
1 1
5
3
-> The following rectangle is not a square
3 2
7
3
-> The following rectangle is a square
3 2
3
3
-> Drawing rectangle:
3 2
7
3
+-----+
|     |
+-----+
-> Drawing rectangle:
3 2
3
3
+-+
| |
+-+
```

- ✓ You may not use global variables.
- ✓ You can write additional functions as long as they are called by the ones on the list above or used in your testing program (i.e., you cannot change the arguments or return type of the functions required, but you could write extra helper functions).