Assignment 6

Class Rectangle: (10 points)

Create a class named 'Rectangle' with the following attributes and methods (sample run for each method included):

1) Each instance should have an x, y, width, and height attributes.

2) You should be able to pass the attributes when creating the rectangle as follows, (where x=5, y=10, width=50, height=100 in that order):

```
r = Rectangle(5, 10, 50, 100)
```

3) Create a method that returns the rectangle as a string (hint: implement __str__). For a rectangle object with attribute values x=5, y=10, width=50, height=100, it should return the string “Rectangle(5, 10, 50, 100)”.

```
>>> r2 = Rectangle(5, 10, 50, 100)
>>> print(r2)
 Rectangle(5, 10, 50, 100)
```

4) Create a method called ``right`` that gets the value of the right edge of the rectangle. It should take no arguments:

```
>>> r3 = Rectangle(3, 5, 10, 20)
>>> r3.right()
 13

>>> r4 = Rectangle(12, 10, 72, 35)
>>> r4.right()
 84
```

5) Create a method called ``bottom`` that gets the value of the bottom edge of the rectangle.

```
>>> r5 = Rectangle(5, 7, 10, 6)
>>> r5.bottom()
 13

>>> r5.y += 12
>>> r5.bottom()
 25
```

6) Create a method called ``size`` that returns the width and height of your rectangle.

```
>>> r6 = Rectangle(1, 2, 3, 4)
>>> r6.size()
 (3, 4)
```

7) Create a method called ``position`` that returns the x and y coordinates of your rectangle.

```
>>> r6.position()
 (1, 2)
```

8) Create a method called ``area`` that returns the area of your rectangle.

```
>>> r6.area()
 12
```

9) Create a method called ``expand`` that takes an offset value and returns a copy of the rectangle expanded with offset in all directions.

```
>>> r = Rectangle(30, 40, 100, 110)
>>> print(r)
 Rectangle(30, 40, 100, 110)

>>> r1 = r.expand(offset=3)
>>> print(r1)
 Rectangle(27, 37, 106, 116)

The original rectangle should not be modified.
 >>> print(r)
 Rectangle(30, 40, 100, 110)

Negative values should return a shrunken rectangle.
>>> print(r.expand(-5))
 Rectangle(35, 45, 90, 100)
```

Part 2: (5 points)

Create a method called ``contains_point`` that takes coordinates x and y as parameters and returns True if the point in inside (or on the edge) of the rectangle and False otherwise.

```
>>> r = Rectangle(30, 40, 100, 110)

>>> r.contains_point(50, 50)
  True

>>> r.contains_point(30,40)
  True

>>> r.contains_point(130, 150)
  True

>>> r.contains_point(131, 50)
  False

>>> r.contains_point(0,0)
  False
```