ICS Fall 2021
Assignment 5

Please note: For some questions, starting codes are available. Please fill in blanks in starting codes. For questions without starting codes, you can write the solutions in any form you like. Since we use an autograder to grade on your code, please strictly name the functions and the submitted files as required by the questions. Moreover, when you submit your answers, you should zip the files, not the whole folder of your answers. Lastly, the starting codes may contain some tests for you to check if your code is correct.

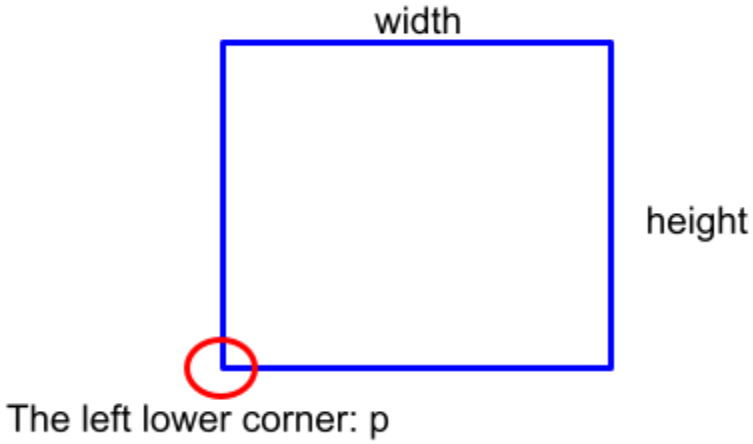
Exercise 1 - Rectangle (Please complete the `rectangle_student.py`)

The following code defines a `Polygon class`, which uses the `Point class` defined in `point.py`.

```python
from Point import Point
import matplotlib.pyplot as plt


class Polygon:
    """
    A polygon class with a list of points
    """

    def __init__(self):
        self.points = []

    def add_point(self, x, y):
        self.points.append(Point(x, y))

    def get_point(self, index):
        #check that the index is valid
        if 0 < index < len(self.points):
            return self.points[index-1]
        else:
            return

    def plot(self):
        x = []
        y = []
        for i in range(len(self.points)):
            x.append(self.points[i].x)
            y.append(self.points[i].y)
        x.append(self.points[0].x)
        y.append(self.points[0].y)
        plt.plot(x, y)
        plt.show()
        return
```

Write a `Rectangle` class which is a subclass of `Polygon`. However, the Rectangle is initialized by the left lower corner which is an instance of the

`Point` class, the width and height , which is different to that of `Polygon`.



The `__init__()` of Rectangle should look like the following:

```
    def __init__(self, width, height, p):
```

You need to find a way to initialize the `Rectangle` properly so that the methods it inherits from the `Polygon` can work without any modification.

Exercise 2 - Permutations

Write a function called `permute(nums)` which takes a list of integers and returns all possible permutations. For example, if `nums = [1, 2, 3]`, then, `permute(nums)` returns:

```
[[1, 2, 3], [1, 3, 2], [2, 1, 3], [3, 1, 2], [2, 3, 1], [3, 2, 1]]
```

Hint: You may use recursion to solve this problem.