INDIAN INSTITUTE OF TECHNOLOGY PALAKKAD

COMPUTER SCIENCE AND ENGINEERING
Indian Institute of Technology, Palakkad
CS5107: Programming Lab
Lab 2: All about classes

Time: 1 week | Max points: 100

1. Some instructions for you codes:
   - You are expected to write codes from scratch without importing any packages.
   - Codes should be compatible with *Python3*.
   - Code for each question should be placed in separate file (*Q2.py, Q3.py, and Q4.py*).
   - Codes should be properly commented.

2. Create a class `Polygon` with attributes `numSides` and `area` such that [30]
   - It should be possible to create an object of this type by specifying the number of sides and area of the polygon.

```
p = Polygon(10, 23)
```

   - A polygon should have at-least 3 sides and non-zero area. Else, an exception should be thrown and handled.

```
p = Polygon(1, 23)
```

Expected output:

```
<class 'Exception'>
Number of sides should be atleast 3
```

```
p = Polygon(10, -23)
```

Expected output:

```
<class 'Exception'>
Polygon should have postive area
```

   - It should be possible to `print` an object of type `Polygon`.

```
p = Polygon(13, 123)
print(p)
```

Expected output:

```
Polygon with 13 sides and area 123
```

Continued on next page...

3. Create a derived class `Triangle` from the base class `Polygon` such that [20]

   - `Triangle` class has no additional attributes.
   - It should be possible to create an object of type `Triangle` by specifying the length of each side.

     ```
     t = Triangle(3, 4, 5)
     ```

   - Sides of a triangle should satisfy certain property. Else, an exception should be thrown and handled.

     ```
     t = Triangle(1, -2, 3)
     ```

     Expected output:

     ```
     <class 'Exception'>
     Triangle should have postive side-lengths
     ```

     ```
     t = Triangle(3, 2, 1)
     ```

     Expected output:

     ```
     <class 'Exception'>
     Side-lengths do not form a triangle
     ```

   - It should be possible to `print` an object of type `Triangle`.

     ```
     t = Triangle(3, 4, 5)
     print(t)
     ```

     Expected output:

     ```
     Triangle with area 6
     ```

Continued on next page...

4. ***Note**: To solve this question, you may need to modify the `Polygon` class created in Q2.*

   Create a class `Paper` that has an attribute `area` (if needed, you can define additional attributes) such that [50]

   - It should be possible to create an object of type `Paper` by specifying area of the paper.

   ```
   pap = Paper(200)
   ```

   - Paper should have a positive area. Else, throw an exception and handle it.

   ```
   pap = Paper(-1)
   ```

   Expected output:

   ```
   <class 'Exception'>
   Paper should have a positive area
   ```

   - Overload the + operator so that you will be able add a *polygon or triangle* object to the paper object. Each time an object is added to the paper, available area of the paper decrease by the area attribute of the object. If the available area is less than the area of the polygon being added, an exception should be thrown and handled.

   ```
   pap = Paper(200)
   p = Polygon(10, 100)
   pap = pap + p
   pap = pap + Polygon(5, 45)
   print(type(pap).__name__)
   ```

   Expected output:

   ```
   Paper
   ```

   ```
   pap = Paper(200)
   p1 = Polygon(10, 100)
   p2 = Polygon(20, 150)
   pap = (pap + p1) + p2
   ```

   Expected output:

   ```
   <class 'Exception'>
   Paper does not have sufficient free area to fit the polygon
   ```

Question 4 continues on the next page...

- It should be possible to `print` an object of type `Paper`; outputs free area in the paper and details of all polygons and triangles added to the paper.

```
pap = Paper(200)
pap = pap + Polygon(10, 100)
pap = pap + Polygon(20, 50)
pap = pap + Triangle(3, 4, 5)
print(pap)
```

Expected output:

```
Paper has free area 44.0 out of 200, and contains:
Polygon with 10 sides and area 100
Polygon with 20 sides and area 50
Triangle with area 6.0
```

- Polygons with 3 sides added to the paper are always stored as triangles.

```
pap = Paper(200)
pap = pap + Polygon(3, 100)
print(pap)
```

Expected output:

```
Paper has free area 100 out of 200, and contains:
Triangle with area 100
```

- It should have a method `merge` that will combine polygons with same number of sides together. Area of the resultant polygon is sum of areas of the individual polygons.

```
pap = Paper(2000)
pap = pap + Polygon(10, 100)
pap = pap + Polygon(20, 50)
pap = pap + Polygon(10, 200)
pap = pap + Polygon(3,24)
pap = pap + Triangle(3, 4, 5)
print(pap)
pap.merge()
print(pap)
pap = pap + Polygon(10, 123)
print(pap)
```

Question 4 continues on the next page...

Expected output:

```
Paper has free area 1620.0 out of 2000, and contains:
Polygon with 10 sides and area 100
Polygon with 10 sides and area 200
Polygon with 20 sides and area 50
Triangle with area 24
Triangle with area 6.0
Paper has free area 1620.0 out of 2000, and contains:
Polygon with 10 sides and area 300
Polygon with 20 sides and area 50
Triangle with area 30.0
Paper has free area 1497.0 out of 2000, and contains:
Polygon with 10 sides and area 300
Polygon with 10 sides and area 123
Polygon with 20 sides and area 50
Triangle with area 30.0
```

- It should have a method `erase` that will remove everything from the paper.

```
pap = Paper(200)
pap = pap + Polygon(10, 100)
pap = pap + Polygon(20, 50)
pap = pap + Triangle(3, 4, 5)
print(pap)
pap.erase()
print(pap)
```

Expected output:

```
Paper has free area 44.0 out of 200, and contains:
Polygon with 10 sides and area 100
Polygon with 20 sides and area 50
Triangle with area 6.0
Paper has free area 200 out of 200, and contains:
```