# Introduction to Programming

Labs – Week 12b

## Exercise 1

Write a data type **`Point`** that implements the following API

| `public class Point` | |
|---|---|
| `Point(double x, double y)` | |
| `double distanceTo(Point q)` | *Euclidean distance between this point and q* |
| `String toString()` | *string representation* |

Following is an example of a test client.

```
public static void main(String[] args) {

      Point p = new Point(1,2);
      Point origin = new Point(0,0);

      System.out.println("Point is:" + p);
      System.out.println("Distance to origin is:" p.distanceTo(origin))

}
```

## Exercise 2

Add the following method to the **`Point`** class:

```
public int quadrant()
```

Returns which quadrant of the $x/y$ plane this **`Point`** object falls in. Quadrant 1 contains all points whose $x$ and $y$ values are both positive. Quadrant 2 contains all points with negative $x$ but positive $y$. Quadrant 3 contains all points with negative $x$ and $y$ values. Quadrant 4 contains all points with positive $x$ but negative $y$. If the point lies directly on the $x$ and/or $y$ axis, return $0$.

## Exercise 3

Add the following method to the **`Point`** class:

```
public void flip()
```

Negates and swaps the $x/y$ coordinates of the **`Point`** object. For example, if the object initially represents the point $(5, -3)$, after a call to **`flip`**, the object should represent $(3, -5)$. If the object initially represents the point $(4,17)$, after a call to **`flip`**, the object should represent $(-17, -4)$.

## Exercise 4

Use ArrayList to solve the following: https://leetcode.com/problems/decompress-run-length-encoded-list/

## Exercise 5

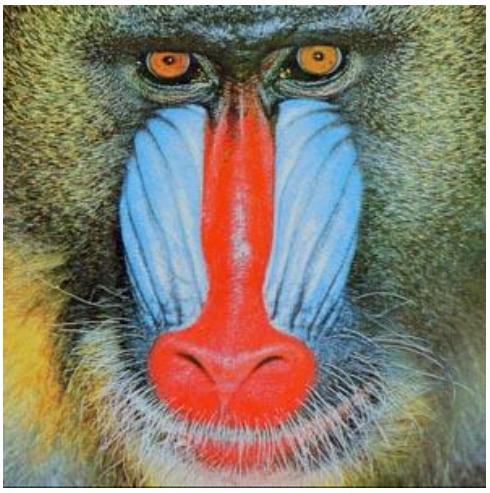
Write a program **`Rotation.java`** that takes two command-line arguments (the name of an image file and a real number **`r`** and rotates the image **`r`** degrees counterclockwise. To rotate, set the color of each pixel $(t_i, t_j)$ in the target image from a source pixel $(s_i, s_j)$ whose coordinates are given by the following formulas:

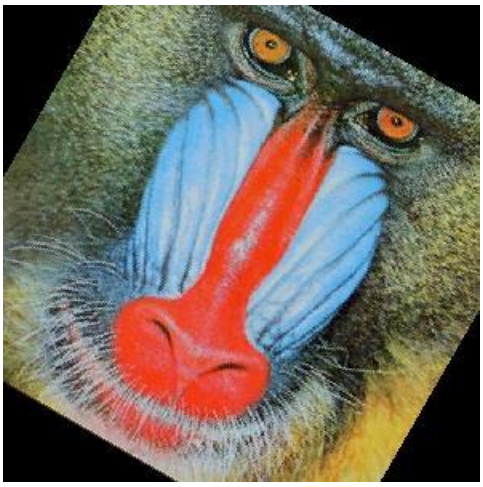
$$s_i = (t_i - c_i)\cos r - (t_j - c_j)\sin r + c_i$$

$$s_j = (t_i - c_i)\sin r + (t_j - c_j)\cos r + c_j$$

where $(c_i, c_j)$ is the center of the image.

*Note:* **`Math.sin()`** and other trigonometric function assumes angle in radians. To convert angle in degrees to angle in radians, use **`Math.toRadians()`** method.





## Exercise 6

Creating a swirl effect is similar to rotation, except that the angle **`r`** changes as a function of distance to the center of the image. Use the same formulas as in the previous exercise, but compute **`r`** as a function of $(s_i, s_j)$, specifically $\frac{\pi}{256}$ times the distance to the center.