Department of Statistics
STATS 782 Statistical Computing
Assignment 4 (2019FC)

Total: 60 marks (including the Bonus) **Due: 3:00 pm, Wed 5 June 2019**

- Submit a hard copy with a signed, QR-coded coversheet to the SRC and a soft copy of your R code on Canvas. Other instructions might be posted on the class webpage. Ideally, the soft copy is an `.Rmd` or `.Rnw` file and the hard copy is produced by 'knitting' the file.
- Include everything in the hard copy: R code (tidied up), outputs (including error/warning messages), and your explanations (if any).
- Print some intermediate results to show how your code works step by step, if not obvious.
- Comment your code wherever appropriate, e.g., for functions, blocks of code, and key variables.
- The marker may run your R code. So include your name and ID on all files. The file names should contain your UPI. Software such as RMarkdown and knitr are recommended.
- Your mark for this assignment will depend on getting the right answer, the elegance/efficiency of your approach, and the tidiness and documentation of your code/report. The R Google Style is recommended. **Marks (up to 7) will be deducted for messy code, etc.**
- Any "**Bonus** exercises" are optional; it is possible to get full marks by completing only the compulsory parts of the assignment *perfectly*. Your mark can be improved (up to the maximum) by successfully attempting extension exercises.
- Look at the PDF version of this file as it may contain colour.

1. [**30 marks**] This question uses R S3 OOP to manipulate regular polygons whose vertices are (by default) on the unit circle of the plane, i.e., centred always at the origin.

   Write some code to implement the following. You'll need to define a class called `"regpolygon3"` that contains information on its radius and angle because later we allow them to change. To keep things uniform, by default, one of the vertices is located at (0, 1). See Figure 1.

   (a) Write a constructor function to create a regular polygon with any number of sides, from 3 and upwards. Allow for circles. Hence each object should contain the following information: the number of sides, the coordinates of one of its vertices, and an optional name such as `"triangle"`, `"square"`, `"pentagon"`, `"hexagon"`,. . . , `"circle"`.

   Build in some error checking too. For example, the number of sides should be integer-valued, the radius should be positive and finite, etc. Of course, missing values are not allowed. Run your code to create something like the following. [5 marks]

```
rpg3 <- newregpolygon3()  # Default is an equilateral triangle
rpg4 <- newregpolygon3(sides = 4)  # Square
rpg8 <- newregpolygon3(sides = 8)  # Octagon
rpgInf <- newregpolygon3(sides = Inf)  # Circle
```

```r
bad1 <- newregpolygon3(sides = 5.5)  # Give an informative error msg
```

(b) Write three generic R S3 accessor functions to return

- the number of sides of the regular polygon,
- the radius, and
- all the vertices (as a 2-column matrix, and for circles, have c.100 rows as an approximation.

Run your code as follows: [5 marks]

```r
sides(rpg8)
radius(rpg8)
vertices(rpg8)
```

(c) Write a print methods function to display the objects. The following commands should print out something appropriate, i.e., a short self-contained description of the object, including the coordinates of at least one vertex. [3 marks]

```r
rpg3
rpg4
rpg8
print(rpg8)  # Same as previous line
```

(d) Now for plotting. Write some R S3 code so that, for example,

```r
> plot(rpg3, rpg4, rpg8, border = 1:3, lty = 1:3, lwd = c(2, 2, 3))
```

produces something like Figure 1. Of course, any number of regular polygons should be handled. Try to get it so that the aspect ratio is unity, i.e., x and y axes are the same length no matter what shape the window is. Test your code on the above example to obtain Figure 1. [7 marks]
Hints:

- the first two arguments of your plotting function should be "`object, ...`" because `object` is to be plotted and `...` represents optional regular polygons.
- Put the `...` into a list and plot each component of the list.

(e) Now we want to allow for 'addition' and 'multiplication'. For example,

```r
0.5 + rpg3
rpg3 * 3
```

The first means to rotate the regular polygon *clockwise* by 0.5 radians. The second statement means to multiply the radius of the encasing circle by 3. Of course,

```r
0.5 + 3 * rpg3
3 * (0.5 + rpg3)
```

are okay too and should give the same answer (why?).
Test your code with:

(i)
```r
0.5 + 3 * rpg3
3 * (0.5 + rpg3)
```

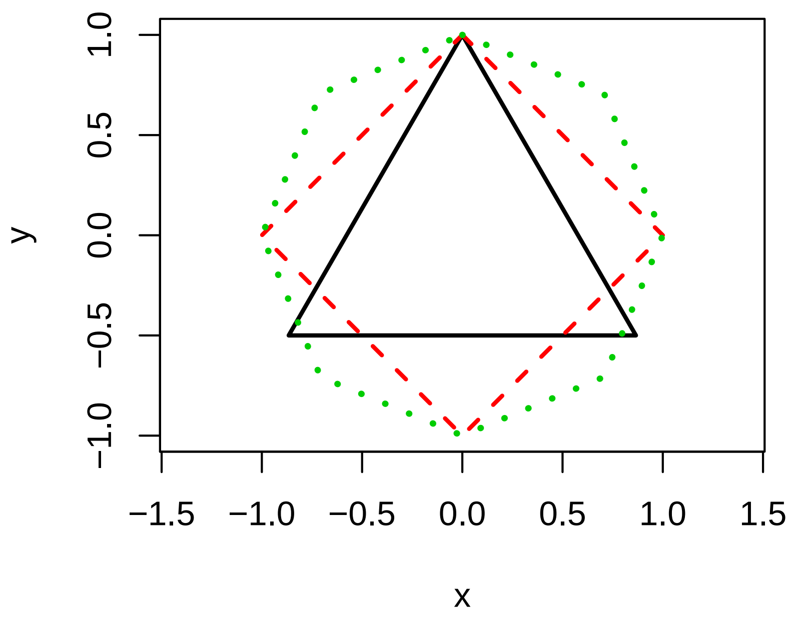

Figure 1: Output from plotting some regular polygons using S3.

(ii)
```r
plot(rpg3 + pi/2, 2 * rpg4 + pi/4, border = c(2, 4), lty = 1:2, lwd = 2)
```

Display your plots. [10 marks]

(f) **Bonus**. Use your function to create the Star of David. Include an “eye-catching visual ‘trick’” to enhance your plot. [Bonus: 3 marks]

2. [**30 marks**] Here, we will repeat Question 1 using S4 R OOP, as follows.

   (a) Repeat Question 1(a) but call your class `"regpolygon4"`.

   (b) Repeat Question 1(b).

   (c) Repeat Question 1(c).

   (d) Repeat Question 1(d).

   (e) Repeat Question 1(e).