**CIS 223** **Due: 09/19/2019**

**Group Assignment 1**

---

# Random experiments, Polygon Objects, Fraction Objects

Answer the following questions in a Word (or other word-processing) file. Copy the questions into your document so that I have context for your answers. Title and label axes of all graphs, if graphs are required. Please compress your answer file and code into one zip file and submit it to D2L Brightspace assignment folder.

**Part A:**

The birthday paradox says that the probability that two people in a room will have the same birthday is more than half, provided n, the number of people in the room, is more than 23. This property is not really a paradox, but many people find it surprising. Design a Python program that can test this paradox by a series of experiments on randomly generated birthdays, which test this paradox for n = 5, 6, 7, …, 50. To achieve a reasonably accurate probability, I suggest you repeat at least 1000 times for each n value.

**Submit**: code, data, and graph (about n's and probabilities)

**Part B:**

Design and implement classes/subclasses.

1) Develop an inheritance hierarchy based upon a Polygon class that has abstract methods area( ) and perimeter( ).
2) Implement classes Triangle, Quadrilateral, Pentagon, Hexagon, and Octagon that extend this base class, with the obvious meanings for the area( ) and perimeter( ) methods.
3) Also implement classes, Isosceles Triangle, Equilateral Triangle, Rectangle, and Square, that have the appropriate inheritance relationships.
4) Finally, write a simple program that allows users to create polygons of the various types and input their geometric dimensions, and the program then outputs their area and perimeter.

**Submit**: class hierarchy graph, code of classes, test program, outputs.

**Part C:**

What values are returned during the following series of stack operations, if executed upon an initially empty stack? If no value returned, simply put NA.

| Step # | operation | Value returned | Step # | Operation | Value returned |
|---|---|---|---|---|---|
| 1 | `push(5)` | | 10 | `pop()` | |
| 2 | `push(3)` | | 11 | `push(7)` | |
| 3 | `pop()` | | 12 | `push(6)` | |
| 4 | `push(2)` | | 13 | `pop()` | |
| 5 | `push(8)` | | 14 | `pop()` | |
| 6 | `pop()` | | 15 | `push(4)` | |
| 7 | `pop()` | | 16 | `pop()` | |
| 8 | `push(9)` | | 17 | `pop()` | |
| 9 | `push(1)` | | 18 | `pop()` | |

**Part D:**

1. (2 points) Give the Big-O performance of the following code fragment:

```
i = 1
while i < n
  i = i + i
```

2. (2 points) Give the Big-O performance of the following code fragment:

```
for i in range(n):
  for j in range(n*n):
    for k in range(n):
      k = 2 + 2
```