Assignment 1:

Due Date: Friday January 14, 11:59 p.m.

Objectives:

- Proof by Induction
- Asymptotic Notations
- Program Analysis

Submission Instruction

- **Start early**
- You are allowed to work in groups of at most two students. It is okay if you want to work on your own.
- Create one file named `theory.pdf`.
- Create one file name `code.py:` This file contains the code related to question 5. Note: question 5 is not just coding.
- Put the above two files in a folder named **a1_x_y** where x is your username and y is your assignment partner username (e.g., if I were to develop this assignment with my partner having username barn407, then I would name my project as **a1_mrudafshani_barn407**)
- Zip the above folder (e.g., a1_mrudafshani_barn407.zip) and submit the zip file to the appropriate dropbox folder
- Make sure to put your names (as well as your partner's name) at the top of both the theory.pdf and code.py
- **Make sure to have one submission per group**
-

Problems

Q1. Prove by induction that for all n > 1 the following inequality holds:

$$\frac{1}{n+1}+\frac{1}{n+2}+\ldots+\frac{1}{2n}>\frac{13}{24}$$

Q2. Suppose we have n >= 3 lines so that no two lines are parallel and no three lines intersect at a common point. Prove that at least one of the regions they form is a triangle.
Hint: You can use induction to prove that. The base case is the case where n=3.

Q3. For each of the following pairs of functions f(n) and g(n), either f(n) = O(g(n)) or g(n) = O(f(n)), but not both. Determine which is the case.

a) **$f(n) = (n^2 - n)/2$, $g(n) = 6n$**
b) **$f(n) = n + 2\sqrt{n}$, $g(n) = n^2$**

c) **f(n) = n + log n, g(n) = n √ n**
d) **f(n) = n^2 + 3n + 4, g(n) = n^3**

Q4. Prove the following:

$$2n + 1 = O(2^n)$$

Is this the best upper-bound for the function on the left? In other words, is it a tight upper bound? If not, specify the tight upper bound?

Q5. For each of the following six code segments:

a) Give Big-O analysis of the running time
b) Run the code and give the running time for several values of n
c) Compare your analysis with the actual runtimes obtained

1)

```
sum = 0
for i in range(n):
    Sum += 1
```

2)

```
sum = 0
for i in range(n):
    for j in range(n):
        sum += 1
```

3)

```
sum = 0;
for i in range(n):
    for j in range(n*n):
        sum += 1
```

4)

```
sum = 0
for i in range(n):
    for j in range(i):
        sum += 1
```

5)

```
sum = 0
for i in range(n):
    for j in range(i*i):
        for k in range(j):
            sum += 1
```

6)

```
sum = 0
for i in range(n):
    for j in range(i*i):
        if (j%i == 0):
            for k in range(j):
                sum += 1
```

Make sure you use a large number of values for n to see the effect of input size on the running time of an algorithm