DSC 40B - Discussion 01

Problem 1.

What is the time complexity of the following functions? State your answer using Θ notation.

a)
```
def foo(n):
    for i in range(n**2 - 2*n + 100):
        j = 0
        while j < n:
            j += 1
```

Solution: $\Theta(n^3)$

b)
```
def foo(n):
    while n > 1:
        n /= 10
        print(n)
```

Solution: $\Theta(\log n)$

c)
```
def foo(n):
    for i in range(n):
        for j in range(i**2): # <-- notice the bound!
            print(i + j)
```

Solution: $\Theta(n^3)$

d)
```
def pairs(numbers):
    result = []
    for x in numbers:
        for y in numbers:
            result.append((x, y))

    return result
```

Solution: $\Theta(n^2)$

e)
```
def foo(numbers):
    for pair in pairs(numbers):
        print(sum(pair))
```

Solution: $\Theta(n^2)$

Note that the result of pairs(numbers) is actually only computed once, on the first iteration. On this first iteration, Python will try to produce the 0th element of pairs(numbers), which it will need to compute in $\Theta(n^2)$ time. After this result is computed, subsequent executions of the for loop line will take $\Theta(1)$ time as they simply produce the next element of the precomputed result. So, this function is equivalent to:

```python
def foo(numbers):
    lst = pairs(numbers) # θ(n^2)
    for pair in lst: # θ(n^2)
        print(sum(pair)) # θ(1)
```

Problem 2.

Let $f(n) = \sum_{p=0}^{n} 3^p$. What is f in Θ notation?

Solution:

General form of a geometric sum $\displaystyle\sum_{p=0}^{n} x^p = \frac{1 - x^{n+1}}{1 - x}$.

Substituting our equation yields $\displaystyle\sum_{p=0}^{n} 3^p = \frac{1 - 3^{n+1}}{1 - 3}$.

Therefore, $\boxed{f(n) = \Theta(3^n)}$ after throwing out the constants.