CS 213: Data Structures and Algorithms **(Due Date: 07/02/21)**

Assignment #2

Instructor: Prof. Sharat Chandran

Problem 1. Composite recurrence
Consider two functions f(int n), g(int n) defined as:

```
def f(int n):
    if n == 1:
        return 1
    return f(n-1) + g(n-1)
def g(int n):
    if n == 1:
        return 1
    if (n%2 == 0):
        return g(n/2)
    return g((n+1)/2)
```

What is the big-Oh complexity of `f(n)`?

Problem 2. More Recurrence
Let $T(n) = T(\frac{n}{4}) + T(\frac{n}{2}) + 1$

Take the base case as $T(1) = 1$ and you can assume n to be an even power of 2 so that the inputs to T are always integers.
Find $T(n)$ in terms of Ω notation.

Problem 3. Transformation for recurrences
Let

$$T(n) = 2T(\sqrt{n}) + c$$

and the base case:

$$T(2) = T(1) = 1$$

Give a bound on $T(n)$ in terms of Θ notation. Prove how you obtained the bound.
For simplicity you may consider n to be of a form that ensures only integers are found when you unroll the recursion (that is the inputs to T are always integers).

Problem 4. Master Theorem
Consider the following recurrences:

i $T(n) = 4T(\frac{n}{2}) + n^2 \log^4 n$

ii $T(n) = T(\frac{n}{2}) + \tanh n$

iii $T(n) = T(\frac{n}{2}) + n(2 - \cos n)$

The base case for each of these is $T(1) = \Theta(1)$
For each of these, state whether the master theorem is applicable or not. If yes, state to which of the three cases the recursion belongs to and find the asymptotic bound. If not, state reasons why the theorem is not applicable. In the cases where master theorem is not applicable, can you find the asymptotic bound using other methods?(this is not necessary but may fetch you bonus marks)

Problem 5. I hate loops

```
1: int a = 0
2: for i = 1; i ≤ n; i++ do
3:     for j = i; j ≤ n; j+=i do
4:         a++;
```

Find the asymptotic complexity of the above code in terms of n