Homework Assignment #2
Due: January 30, 2023 at 7:00 p.m.

1. This question is about converting a p-bit number represented in binary into a decimal representation of the number.

First, observe that we can multiply two n-digit decimal numbers to get a decimal representation of the product in $O(n^{\log_2 3})$ time using essentially the same algorithm described in lecture for multiplying numbers in binary. (There was nothing special about the fact that that multiplication algorithm used binary representation.) Call this algorithm DecimalMultiply.

We can add two n-digit decimal numbers to get a decimal representation of the sum in $O(n)$ time using the algorithm that you learned in grade 1. Call this algorithm DecimalAdd.

In class, we also saw a recursive algorithm to compute powers. Here is a slightly revised version of it for computing the decimal representation of powers of 2.

```
1:  function Power2(p)
2:      precondition: p is a non-negative integer (represented in binary)
3:      if p = 0 then
4:          return 1
5:      else
6:          x ← Power2(⌊p/2⌋)
7:          y ← DecimalMultiply(x, x)
8:          if p is even then
9:              return y
10:         else
11:             return DecimalAdd(y, y)
12:         end if
13:     end if
14:     postcondition: returns decimal representation of 2^p
15: end function
```

[1] **(a)** Give a good upper bound on the number of digits in the number stored in x on line 6? Give your answer in terms of p.

[2] **(b)** Give a recurrence for the running time of Power2(p). Explain why your recurrence is correct.

[2] **(c)** Use the Master Theorem to give a good upper bound on the running time of Power2(p). Express your answer in terms of p using big-O notation.

[5] **(d)** Now, give an efficient divide-and-conquer algorithm to convert a given p-bit binary representation of an arbitrary number into its decimal representation. You do *not* have to give a formal proof of correctness for your algorithm, but you should briefly explain why it is correct.

Hint: Split the given p-bit number x into $x_1 \cdot 2^{\lfloor p/2 \rfloor} + x_0$.

```
function Convert(x, p)
    precondition: x and p are integers (represented in binary) with p ≥ 1 and 0 ≤ x < 2^p
    Fill in this code
    postcondition: output is the decimal representation of x
end function
```

[2] **(e)** What is the worst-case running time of your algorithm in part (d)? State your answer in terms of p using big-O notation. Briefly justify your answer.