These exercises are intended to give you some practice proving bounds on recurrences, and proving correctness of recursive programs.

1. Examine the recurrence $R(n)$ below.

$$R(n) = \begin{cases} 
0 & \text{if } n = 1 \\
3n + 3R([n/3]) & \text{if } n > 1
\end{cases}$$

Assume that for all $k \in \mathbb{N}$, $R(3^k) = k3^k$.

**Sample solution:** Prove that $R \in O(n \log n)$. Define $n^* = 3^{\lceil \log_3 n \rceil}$. Then we have:

$$\lceil \log_3 n \rceil - 1 < \log_3 n \leq \lceil \log_3 n \rceil \Rightarrow n^*/3 < n \leq n^*$$

I will also use the assumption (proved last week) that $R$ is nondecreasing.

Let $d = 6$. Then $d \in \mathbb{R}^+$. Let $B = 3$. Then $B \in \mathbb{N}^+$. Let $n$ be an arbitrary natural number no smaller than $B$. Then

$$R(n) \leq R(n^*) \quad \# \text{ since } R \text{ nondecreasing and } n^* \geq n$$

$$= n^* \log_3 n^* \quad \# \text{ by assumption}$$

$$\leq 3n \log_3 (3n) \quad \# n > n^*/3 \Rightarrow 3n > n^*$$

$$= 3n(\log_3 (n + 1) \leq 3n(\log_3 n + 1) \log_3 n) \quad \# n \geq B \geq 3 \Rightarrow \log_3 n \geq 1$$

$$= 6n \log_3 n \leq dn \log_3 n \quad \# d = 6$$

So $R \in O(n \log n)$, since $\log_3 n$ differs from $\log n$ by a constant factor.

**Sample solution:** Prove that $R \in \Omega(n \log n)$. Define $n^* = 3^{\lceil \log_3 n \rceil}$. Then we have:

$$\lceil \log_3 n \rceil - 1 < \log_3 n \leq \lceil \log_3 n \rceil \Rightarrow n^*/3 < n \leq n^*$$

I will also use the assumption (proved last week) that $R$ is nondecreasing.

Let $d = 1/6$. Then $d \in \mathbb{R}^+$. Let $B = 9$. Then $B \in \mathbb{N}$.

Let $n$ be an arbitrary natural number no smaller than $B$. Then

$$R(n) \geq R(n^*/3) \quad \# \text{ since } R \text{ nondecreasing and } n > n^*/3$$

$$= n^*/3 \log_3 n^*/3 \quad \# \text{ by assumption}$$

$$\geq n/3 \log_3 (n/3) \quad \# n^* \geq n \Rightarrow n^*/3 > n/3$$

$$= n/3(\log_3 (n - 1) = n/3 \log_3 n - n/3 = n/6 \log_3 n + n/6 \log_3 n - n/3$$

$$\geq n/6 \log_3 n \quad \# n \geq 9 \geq B \Rightarrow n/6 \log_3 n \geq n/3$$

$$= dn \log_3 n \quad \# d = 1/6$$
So \( R \in \Omega(n \lg n) \), since \( \log_3 n \) differs from \( \lg n \) by a constant. ■

2. Read over the code for `decimal_to_binary` below:

```python
def decimal_to_binary(n: int) -> str:
    
    Return binary string representing n.

    precondition: n is a natural number.

    >>> decimal_to_binary(0)
    '0'
    >>> decimal_to_binary(5)
    '101'

    postcondition: returns binary string representing
    n with no leading zeros (except if n == 0).
    
    if n < 2:
        return str(n)
    else:
        return decimal_to_binary(n // 2) + decimal_to_binary(n % 2)
```

Use the technique from week 7 notes to prove that the precondition implies termination and the postcondition, or find a counter-example.

**Sample solution:** Let \( n \in \mathbb{N} \) and bits \( b_0, \ldots, b_k \in \{0, 1\} \) be such that \( n = \sum_{i=0}^{k} 2^i b_i \). I will use the identities:

\[
|n/2| = \sum_{i=1}^{k} 2^{i-1} b_i \quad \text{and} \quad n \equiv b_0 \mod 2
\]

Define \( P(n) \) : "If \( n \) is a natural number, then `decimal_to_binary(n)` terminates and returns the binary string representing \( n \) with no leading zeros, except if \( n = 0 \)."

I will use complete induction to prove \( \forall n \in \mathbb{N}, P(n) \).

**Inductive step:** Let \( n \in \mathbb{N} \). Assume \( \bigwedge_{j=0}^{n-1} P(j) \). I will show that \( P(n) \) follows.

**Case** \( n < 2 \): If \( n < 2 \) the "if" branch executes, and \( \text{str}(n) \) is returned: "0" if \( n = 0 \) and "1" if \( n = 1 \), which are the binary strings representing 0 and 1, respectively, which can be verified by evaluating the sum \( 0 = \sum_{j=0}^{0} \) and \( 1 = \sum_{j=0}^{1} \). So \( P(n) \) holds in this case.

**Case** \( n \geq 2 \): We have \( 0 \leq n/2 \leq n/2 < n \), so we know \( P(n/2) \land P(n/2) \) by the IH. Let bits \( b_0, \ldots, b_k \in \{0, 1\} \) be such that \( n = \sum_{i=0}^{k} 2^i b_i \), then (by the identity above):

\[
2(|n/2|) + b_0 = 2 \sum_{i=1}^{k} 2^{i-1} b_i + b_0 = \sum_{i=0}^{k} 2^i b_i = n
\]

so the binary string representing \( n \) is the concatenation of the strings for \( b_1, \ldots, b_k \) (representing \( n/2 \), so returned by `decimal_to_binary(n/2)` by the IH) with the string for \( b_0 \) (representing \( n/2 \), so returned by `decimal_to_binary(n/2)` by the IH), which is what is returned by the "else" branch. ■