Question 1. Define the predicates

$$P(n)\colon \text{ For any set } A, \text{ if } |A| = n \text{ then } |\mathcal{P}(A)| = 2^n$$

$$Q(A,n)\colon |A| = n \Longrightarrow |\mathcal{P}(A)| = 2^n$$

a) Prove $\forall n \in \mathbb{N}, P(n)$.

Proof. **Base Case.** To show $P(0)$, consider any set A such that $|A| = 0$. Then $A = \varnothing$ and its only subset is $\varnothing$. Thus $\mathcal{P}(A) = \{\varnothing\} \implies |\mathcal{P}(A)| = 1 = 2^0$, verifying that $P(0)$ is true.

Induction Step. Suppose $P(k)$ holds for some $k \in \mathbb{N}$. Now $P(k+1)$ will be proven to hold. Let A be a set such that $|A| = k+1$. $k+1$ is at least 1, so A possesses at least one element, which will be denoted as a.

Consider the set $A \setminus \{a\}$. Since $\big|A \setminus \{a\}\big| = k$, by the induction hypothesis,

$$\big|\mathcal{P}(A \setminus \{a\})\big| = 2^k$$

Notice that $\mathcal{P}(A \setminus \{a\})$ contains all the subsets of A that do not contain a. The remaining subsets must all contain a. The remaining subsets of A can be obtained by taking every individual element in $\mathcal{P}(A \setminus \{a\})$ and unioning it with $\{a\}$. Thus A contains twice as many subsets as $A \setminus \{a\}$. In mathematical terms,

$$|\mathcal{P}(A)| = 2 \cdot |\mathcal{P}(A \setminus \{a\})| = 2 \cdot 2^k = 2^{k+1}$$

It has been shown that $P(k+1)$ holds.

By the principle of simple induction, $\forall n \in \mathbb{N}, P(n)$.

□

b) Prove that for every set A, $\forall n \in \mathbb{N}$, $Q(n)$. This method does not work. Here is the attempt at the proof:

Proof. Fix a set A. Proceed with using simple induction.

Base Case. Let $n = 0$. To show $Q(A,n)$ holds, suppose that $|A| = 0$. Then $A = \varnothing$. Thus $\mathcal{P}(A) = \{\varnothing\} \implies |\mathcal{P}(A)| = 1 = 2^0$.

Thus $Q(A, 0)$.

Induction Step. Suppose that $Q(A,k)$ holds for some $k \in \mathbb{N}$. To show $Q(A, k+1)$, suppose $|A| = k+1$. However, this is where the problem arises.

The induction hypothesis cannot be utilised since our assumption requires $|A| = k+1$, while the condition to use the induction hypothesis is $|A| = k$.

Thus the proof by induction cannot be continued.

□

Question 2. Let $n, m \in \mathbb{N}$. Let A, B be arbitrary finite sets of size m and n respectively.

a) How many functions are there with domain A and co-domain B?

It can be shown that the answer to this question is n^m. For the sake of convenience, 0^0 will be defined to be equal to 1.

Proof. Let $m, n \in \mathbb{N}$ and A, B be sets of size m and n, respectively. Recall that a function is well defined if and only if every element in the domain maps to a unique element in the codomain. Now consider the following cases:

If $m = n = 0$, a function is vacuously well defined. In this case, the number of functions is equal to $1 = 0^0$.

If $m = 0, n \neq 0$, a function is also vacuously well defined, so the number of functions is equal to $1 = n^0$.

If $m \neq 0, n = 0$, there is no element in the codomain that an element in the domain can map to. This means that no function can be well defined, and thus there exist $0 = 0^m$ functions.

The last case, when both m and n are positive, will be shown by simple induction on m. For $m \in \mathbb{N}^+$, define the predicate

$$P(m)\colon \forall n \in \mathbb{N}^+, \text{there are } n^m \text{ functions with finite domain of size } m$$
$$\text{and finite co-domain of size } n$$

Base Case. If $m = 1$, the domain of the function contains only one element. This one element can map to n possible elements in the codomain, resulting in n^1 distinct functions. Thus $P(1)$ holds.

Induction Step. Suppose that $P(k)$ holds for some positive integer k. Then the domain A contains at least one element a.

Let $n \in \mathbb{N}^+$. Consider the set $A \setminus \{a\}$, which has k elements. By the induction hypothesis, there are n^k functions with domain $A \setminus \{a\}$ and co-domain B. For every such function, it is possible to expand its domain to include a. Since a can map to n different elements in the codomain, which means that every function with domain $A \setminus \{a\}$ and codomain B induce n distinct functions with domain A and codomain B. Thus there are $n^k \cdot n = n^{k+1}$ fuctions that map from A to B. Therefore $P(k+1)$ holds.

By the principle of simple induction, $P(m)$ is true for all positive naturals m.

$\square$

b) Use part (a) to prove the original statement in Q1 directly without the use of induction.

Proof. Let A be a set such that $|A| = n$. Every subset A' of A can be represented as a unique function f that maps elements of A to $\{0, 1\}$:

$$\text{For any } a \in A, \text{ if } a \in A', f(a) = 1. \text{ Otherwise, } f(a) = 0$$

From the part a), there are 2^n different functions with domain A and co-domain $\{0, 1\}$, which also means that there are 2^n subsets of A, which implies that $\mathcal{P}(A) = 2^n$.

□

Question 3. In propositional logic, you have seen the connectives $\neg, \wedge, \vee, \rightarrow$, and $\leftrightarrow$. Prove using structural induction that any proposition built using these connectives is equivalent to a proposition built only using $\neg, \rightarrow$.

Proof. The proof will be done using structural induction.

For a proposition P built using the connectives $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$, Define the predicate

$$Q(P) : P \text{ is equivalent to some proposition built only using } \neg,\ \rightarrow$$

Base Case. For all $P_i(x_{j_1}, x_{j_2}, \dots, x_{j_k})$, they are equivalent to a proposition built only using $\neg, \rightarrow$, which are themselves. Thus $Q(P_i(x_{j_1}, x_{j_2}, \dots, x_{j_k}))$ holds.

Induction Step. Let A, B be propositions such that $Q(A)$ and $Q(B)$ hold. It follows that $A \equiv C, B \equiv D$, where C, D are propositions built from only $\neg,\ \rightarrow$. 5 recursive cases will be considered.

1. $\neg A \equiv \neg C$, thus $Q(\neg A)$ holds

2. $A \rightarrow B \equiv C \rightarrow D$, $Q(A \rightarrow B)$ holds

3. $A \wedge B \equiv C \wedge D$. It will be shown using a truth table that $C \wedge D \equiv \neg(C \rightarrow \neg D)$.

C	D	$C \wedge D$	$\neg(C \rightarrow \neg D)$
T	T	T	T
T	F	F	F
F	T	F	F
F	F	F	F

Since $A \wedge B \equiv \neg(C \rightarrow \neg D)$, which is a proposition built from only $\neg, \rightarrow$. Thus $Q(A \wedge B)$ holds.

4. $A \vee B \equiv C \vee D$. It will be shown using a truth table that $C \vee D \equiv \neg(\neg C \rightarrow D)$.

C	D	$C \vee D$	$\neg(\neg C \rightarrow D)$
T	T	T	T
T	F	T	T
F	T	T	T
F	F	F	F

Since $A \vee B \equiv \neg(\neg C \rightarrow D)$, which is a proposition built from only $\neg, \rightarrow$. Thus $Q(A \vee B)$ holds.

5. $A \leftrightarrow B \equiv C \iff D$. It will be shown using a truth table that $C \leftrightarrow D \equiv \neg((C \rightarrow D) \implies \neg(D \rightarrow C))$.

C	D	$C \leftrightarrow D$	$\neg((C \rightarrow D) \rightarrow \neg(D \rightarrow C))$
T	T	T	T
T	F	F	F
F	T	F	F
F	F	T	T

Since $C \leftrightarrow D \equiv \neg((C \to D) \to \neg(D \to C))$, which is a proposition built from only $\neg, \to$. Thus $Q(A \leftrightarrow B)$ holds.

By the principle of structural induction, $Q(P)$ holds for all propositions P.

□

Question 4. Consider the following two-pointer style Python program which finds whether a given string s is a palindrome or not:

```
def check_if_palindrome(s):
    left = 0
    right = len(s) - 1
    while left < right:
        if s[left] != s[right]:
            return False
        left += 1
        right -= 1
    return True
```

Prove correctness and termination. Clearly state the loop variant and the loop invariant, and use induction properly.

Proof. (Note: / will denote integer division.)

This program takes a string as an input and returns true if the string is a palindrome, and false if the string is not.

First, it will be proven that the program loop terminates. Define the loop variant to be *right*. *right* is initialized to be $\texttt{len}(s) - 1$ and decrements by 1 every loop iteration. If $\texttt{len}(s) = 0$, $left > right$ and the loop never runs. Otherwise, *right* is an integer that is bounded below by $left$ that decreases every loop iteration. Thus the loop will terminate.

Now correctness will be proven. Let s be a string and i be a natural number. The loop invariant will be defined as

$$P(i)\colon (0 \leq i \leq \texttt{len}(s)/2 + 1) \implies (\forall k > 0, k < i \implies s[k-1] = s[\texttt{len}(s) - k])$$

The goal is to show that $P(\texttt{len}(s)/2+1)$ is true after the last iteration of the loop is executed, which shows that the first half of the string is exactly the second half reversed. This will be accomplished by using induction on i to show that for every natural i, $P(i)$ is true at the beginning of the ith iteration.

Base Case. Let $i = 1$. It is quite clear that $0 \leq i \leq \texttt{len}(s) + 1$ is always true. But for every $k \in \mathbb{N}, k > 0$, k can never be less than 1. Therefore $P(1)$ is vacuously true.

Induction Hypothesis. Suppose that for some positive natural i, $P(i)$ is true at the beginning of the ith iteration.

Induction Step. Suppose that $0 \leq i + 1 \leq \texttt{len}(s)/2$. If $i + 1 \in \{0, 1\}$, then $P(i + 1)$ follows trivially. Otherwise, we have that $1 \leq i + 1 \leq \texttt{len}(s)/2 - 1 \implies 0 \leq i \leq \texttt{len}(s)/2 - 2$, which implies the conclusion of the induction hypothesis. To show that $P(i+1)$ is true at the beginning of the $i + 1$th iteration, it suffices to show that when $k = i$, $s[k - 1] = s[\texttt{len}(s) - k]$ after the ith iteration ends, since the values of k less than i are covered by the induction hypothesis.

Notice that $left$ starts at 0 and increments once at the end of every iteration, and *right* starts at $\texttt{len}(s) - 1$ and decrements at the end of every iteration. Thus, at the start of the ith iteration, $left$ and *right* will have been incremented or decremented $i - 1$ times, so $left = i - 1$ and $right = \texttt{len}(s) - i$.

Now, iterating through the loop, there are two cases to consider.

1. $s[left] \neq s[right]$:

   This signifies that the string is not a palindrome and the program terminates by returning false, satisfying the postconditions of the program. The $i+1$th iteration is never reached, so this case can be disregarded

2. $s[left] = s[right]$:

   This implies $s[i-1] = s[\texttt{len}(s) - i]$. Here, the body of the if-statement block does not run and $left$ and $right$ get incremented and decremented.

   Then, the ith iteration ends and the $i+1$th iteration begins. Note that $s[k-1] = s[\texttt{len}(s)-k]$ for all $k < i$, as well as when $k = i$. Thus $P(i+1)$ is true at the start of the $i+1$th iteration.

By the principle of simple induction, $P(i)$ is true at the start of the ith iteration of the loop for all natural numbers i.

□

Question 5. Answer the following:

a) **(2 pt)** For any natural n, how many n-ary connectives are there? You do not need to justify your answer using induction. *Hint: Check Q2.*

Proof. n-ary connectives can be represented as a function from $\{T, F\}^n$ to $\{T, F\}$, so by using question 2, it can be concluded that there are 2^{2^n} different n-ary connectives.

$\square$

b) **(13 pts)** Prove that any truth table that can ever exist is the truth table of some proposition built using only $\{\neg, \wedge\}$.

Proof. For positive naturals n, define the predicate

$P(n)$: Any n-ary truth table is the truth table of some proposition built using only $\{\neg, \wedge\}$

The statement $\forall n \in \mathbb{N}^+, P(n)$ will be proved using simple induction on n.

Base Case. Let $n = 1$. Let p be an arbitrary predicate. Then all the possible unary truth tables are

p	Y
T	T
F	T

p	Y
T	T
F	F

p	Y
T	F
F	T

p	Y
T	F
F	F

which represent the propositions $T, p, \neg p$, and F respectively. Since they are built using only $\{\neg, \wedge\}$, $P(1)$ is true.

Induction Hypothesis. Suppose that for some positive natural k, $P(k)$ is true. Now consider any $n+1$-ary truth table Y with inputs $p_1, p_2, \ldots, p_n, p_{n+1}$. The truth table can be seperated into two k-ary truth tables by fixing p_{k+1} to be either true or false. Label the truth tables by Y_T and Y_F, respectively.

By the induction hypothesis, there are propositions built from $\{\neg, \wedge\}$ such that their truth tables are exactly equal to Y_T and Y_F. These propositions will be denoted by $\Theta_T(p_1, \ldots, p_k)$ and $\Theta_F(p_1, \ldots, p_k)$.

Consider the $k+1$-ary truth function

$$\Theta(p_1, p_2, \ldots, p_k, p_{k+1}) \coloneqq (p_{k+1} \wedge \Theta_T(p_1, \ldots, p_k)) \vee (\neg p_{k+1} \wedge \Theta_F(p_1, \ldots, p_{k+1}))$$

It will be shown that this truth function's truth table is Y.

Let $p_1, p_2, \ldots, p_k, p_{k+1} \in \{T, F\}$. There are two cases:

(a) If p_{k+1} is T, the output of Y is precisely $\Theta_T(p_1, \ldots, p_k)$. Notice that

$$\Theta(p_1, \ldots, p_k, T) \equiv (T \wedge \Theta_T(p_1, \ldots, p_k)) \vee (F \wedge \Theta_F(p_1, \ldots, p_k)) \equiv \Theta_T(p_1, \ldots, p_k)$$

Thus $\Theta(p_1, \ldots, p_k, p_{k+1})$ matches the output of Y when p_{k+1} is true.

(b) If p_{k+1} is F, the output of Y is equivalent to $\Theta_F(p_1,\dots,p_k)$. Note that

$$\Theta(p_1,\dots,p_k,F) \equiv (F \wedge \Theta_T(p_1,\dots,p_k)) \vee (T \wedge \Theta_F(p_1,\dots,p_k)) \equiv \Theta_F(p_1,\dots,p_k)$$

Thus $\Theta(p_1,\dots,p_k,p_{k+1})$ also matches the output of Y when p_{k+1} is false.

Since $\Theta(p_1,\dots,p_k,p_{k+1})$ matches the output of Y both when $p_{k+1}=T$ and $p_{k+1}=F$, it can be concluded that $\Theta(p_1,\dots,p_k,p_{k+1})$ always matches the output of Y.

Using De Morgan's law, it follows that

$$\Theta(p_1,\dots,p_k,p_{k+1}) = (p_{k+1} \wedge \Theta_T(p_1,\dots,p_k)) \vee (\neg p_{k+1} \wedge \Theta_F(p_1,\dots,p_{k+1}))$$

$$= \neg(\neg(p_{k+1} \wedge \Theta_T(p_1,\dots,p_k)) \wedge \neg(\neg p_{k+1} \wedge \Theta_F(p_1,\dots,p_{k+1})))$$

Therefore Θ is a proposition built from only $\{\neg,\wedge\}$, so $P(k+1)$ is true.

By the principle of simple induction, every truth table is the truth table of some truth function.

$\square$