Q1:

a)

The statement is false.

since $1 + b + b^2 + \cdots + b^n = \begin{cases} \frac{1-b^{n+1}}{1-b}, \ if\ b \neq 1 \\ n, \ if\ b = 1 \end{cases}$

consider when $b > 1$, function $f = \frac{1-b^{n+1}}{1-b} = \frac{1}{1-b} - \frac{1}{1-b}b^{n+1} = \theta(b^{n+1})$, the statement is false.

b)

The statement is true.

Assume $f, g: \mathbb{Z}^+ \to \mathbb{R}^{\geq 1}$ satisfies $f = \Theta(g)$, then

$$\forall x > c \in \mathbb{Z}, \qquad k_1 g(x) < f(x) < k_2 g(x)$$

for some positive constants k_1 and k_2. Then clearly

$$2^{k_1 g(x)} < 2^{f(x)} < 2^{k_2 g(x)}$$

$$2^{k_1} \cdot 2^{g(x)} < 2^{f(x)} < 2^{k_2} \cdot 2^{g(x)}$$

Hence

$$\forall x > c \in \mathbb{Z}, \qquad K_1 G(x) < F(x) < K_2 G(x)$$

Where $K_1 = 2^{k_1}$, $K_2 = 2^{k_2}$.

By definition, $F = \Theta(G)$.

c)

The statement is false.

Assume $f, g: \mathbb{Z}^+ \to \mathbb{R}^{\geq 1}$ satisfies $f = \mathrm{o}(g)$, then

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

So

$$\lim_{n\to\infty} \frac{F(n)}{G(n)} = \lim_{n\to\infty} \frac{2^{f(n)}}{2^{g(n)}} = \lim_{n\to\infty} 2^{f(n)-g(n)} = 0$$

Q2:

a)

$$T(n) = 9T\left(\frac{n}{3}\right) + n^2$$

Since $9 = 3^2$, by master theorem, $T(n) = \Theta(n^2 \log_3 n) = \Theta(n^2 \log n)$

b)

$$T(n) = 4T\left(\frac{n}{4}\right) + n\log n$$

By drawing the recursion tree,

$$\begin{aligned}
T(n) &= cn\log n + 4 \times c\left(\frac{n}{4}\log\frac{n}{4}\right) + 4^2 \times c\left(\frac{n}{4^2}\log\frac{n}{4^2}\right) + \cdots + 4^{\log_4 n} \times c\left(\frac{n}{4^{\log_4 n}}\log\frac{n}{4^{\log_4 n}}\right)\\
&= cn\left(\log n + \log\frac{n}{4} + \log\frac{n}{4^2} + \cdots + \log\frac{n}{4^{\log_4 n}}\right)\\
&\le cn(\log n + \log n + \log n + \cdots + \log n)\\
&= cn\log_4 n\log n\\
&\in O(n\log^2 n)
\end{aligned}$$

At the same time,

$$\begin{aligned}
T(n) &= cn\left(\log n + \log\frac{n}{4} + \log\frac{n}{4^2} + \cdots + \log\frac{n}{4^{\log_4 n}}\right)\\
&= cn\log_4 n\log n - cn\left(\log 1 + \log 4 + \log 4^2 + \cdots + \log 4^{\log_4 n}\right)\\
&\ge cn\left(\log 1 + \log 4 + \log 4^2 + \cdots + \log 4^{\log_4 n}\right) \qquad \textit{since } T(n) \ge 0\\
&= cn\left(\log 4\,(0 + 1 + 2 + 3 + \cdots + \log_4 n)\right)\\
&= cn\left(\frac{1}{2}(\log 4)(\log_4 n + 1)(\log_4 n)\right)\\
&\ge \frac{1}{2}cn\log_4 n\log_4 n\\
&\in \Omega(n\log^2 n)
\end{aligned}$$

Therefore $T(n) \in \Theta(n\log^2 n)$.

c)

$$\begin{aligned}
T(n) &= T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right) + n\\
&\le n + \left(\frac{1}{4} + \frac{3}{4}\right)n + \left(\frac{1}{4} + \frac{3}{4}\right)^2 n + \cdots + \left(\frac{1}{4} + \frac{3}{4}\right)^{\log_4 n} n\\
&= n(\log_4 n + 1)\\
&\in \Omega(n\log n)
\end{aligned}$$

$$
\begin{aligned}
&T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right) + n \\
&\geq n + \left(\frac{1}{4} + \frac{3}{4}\right)n + \left(\frac{1}{4} + \frac{3}{4}\right)^2 n + \cdots + \left(\frac{1}{4} + \frac{3}{4}\right)^{\log_{\frac{4}{3}} n} n \\
&= n\left(\log_{\frac{4}{3}} n + 1\right) \\
&\in O(n \log n)
\end{aligned}
$$

So $T(n) \in \Theta(n \log n)$

d)

$$
\begin{aligned}
&T(n) = \sqrt{n} \cdot T\left(\sqrt{n}\right) + n \\
&= n^{\frac{1}{2}} \cdot \left(n^{\frac{1}{4}} \cdot \left(n^{\frac{1}{8}} \cdot \left(n^{\frac{1}{16}} \cdot (\ldots T(1) + \cdots) + n^{\frac{1}{8}}\right) + n^{\frac{1}{4}}\right) + n^{\frac{1}{2}}\right) + n \\
&= \left(n^{\frac{1}{x}} \ldots n^{\frac{1}{16}} n^{\frac{1}{16}} n^{\frac{1}{8}} n^{\frac{1}{4}} n^{\frac{1}{2}}\right) \ldots + \left(n^{\frac{1}{8}} n^{\frac{1}{8}} n^{\frac{1}{4}} n^{\frac{1}{2}}\right) + \left(n^{\frac{1}{4}} n^{\frac{1}{4}} n^{\frac{1}{2}}\right) + \left(n^{\frac{1}{2}} n^{\frac{1}{2}}\right) + n
\end{aligned}
$$

Where x is infinitely large.

$$T(n) = n \times \log_2 x$$

$$T(n) = n \log(\log n)$$

Q3:

The worst case is that it is not a prime, so it goes all of the loops.

For the outer loop, j starts from 2, increment by 1 each time, and ends when $j^2 > n$.

So $j = 2,3,4, \ldots, \lfloor\sqrt{n}\rfloor$, outer loop runs $\sqrt{n} - 1$ times.

For the inner loop, k starts from 2, increment by 1 each time, and ends when $jk > n$.

The worst case runtime

$$
\begin{aligned}
T(n) &= \sum_{j=2}^{\sqrt{n}} \sum_{k=2}^{\frac{n}{j}} 1 \\
&= \sum_{j=2}^{\sqrt{n}} \frac{n}{j} - 1 \\
&= \left(\frac{n}{2} - 1\right) + \left(\frac{n}{3} - 1\right) + \cdots + \left(\frac{n}{\sqrt{n}} - 1\right) \\
&= n\left(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{\sqrt{n}}\right) - n - \left(\sqrt{n} - 1\right) \\
&\in O\left(n \log \sqrt{n}\right)
\end{aligned}
$$

Q4:

1: For all $a, b \in [1, n]$, compute $A[a] + A[b]$ and store the sum into a tree T.

2: For all $a, b \in [1, n]$, compute $B[a] + B[b]$, and search the sum in tree T. Return true if the sum is found. If all sums are not found in the tree, return false.

```
COMMON_SUM(A, B, n){
      //A, B are int arrays of size n.
      //T is an empty tree.
      Tree T;
      for (int x = 0; i < n; ++x){
            for (int y = 0; i < n; ++y){
                  T.insert(A[x] + A[y]);
            }
      }
      for (int x = 0; i < n; ++x){
            for (int y = 0; i < n; ++y){
                  if (T.search(A[x] + A[y])){
                        return true;
                  }
            }
      }
      return false;
}
```

This is a brute force method. The first loop saves every possible sum of elements form array A. In the second loop, for each sum we search it in the previous saved data base: if it is a match, the sum appeared before, so there are two numbers in A sum to the same number as the sum from array B.

The first big loop runs n^2 times. Inside the loop, the insert takes $\Theta(\log(n^2)) = \Theta(2\log(n)) = \Theta(\log(n))$ time, so the insertion takes $\Theta(n^2 \log(n))$ time. Similarly for the second loop, it runs n^2 times, each takes $\Theta(\log(n))$ time for searching in a tree. In total the algorithm takes $\Theta(n^2 \log(n))$ time.