[redacted]

Question 1:

1. The claim is True.
   From the Given we receive that $f(n) = \Theta(g(n)) \implies \exists\, c_1, c_2, n_0 > 0$ such that $\forall n > n_0$ we get that $0 \le c_1 \cdot g(n) \le f(n) \le c_2 \cdot g(n)$
   From the given we also know that $g(n) = \Theta(h(n)) \implies \exists d_1, d_2, n_1 \colon \forall n > n_1$ such that $0 \le d_1 \cdot h(n) \le g(n) \le d_2 \cdot h(n)$

   We want to prove the following: $\exists e_1, e_2, n_2 > 0 \colon \forall n > n_2 \colon 0 \le e_1 h(n) \le f(n) \le e_2 h(n)$

   We will define n_2 such that $n_2 = max\{n_0, n_1\}$: $\forall n > n_2$ the following is true:

   $f(n) \le c_2 \cdot g(n) \le c_2 \cdot d_2 h(n)$
   $f(n) \ge c_1 \cdot g(n) \ge c_1 \cdot d_1 h(n)$

   We will choose $e_1 = min\{c_1, d_1, c_1 \cdot d_1\}$, $e_2 = max\{c_2, d_2, c_2 \cdot d_2\}$, ($c_1, d_1, c_2, d_2$ are all positive constants therefore e_1, e_2 are also bigger than 0) Therefore we will get that $0 \le e_1 h(n) \le f(n) \le e_2 h(n) \implies$ by definition $f(n) = \Theta(h(n))$ and thus the claim is correct. ■
2. The claim is False.
   We will show such with a counter example: $f(n) = n$, $g(n) = \frac{1}{n}$
   $f(n) + g(n) = n + \frac{1}{n} = \mathcal{O}(n)$
   $f(n) \cdot g(n) = n * \frac{1}{n} = 1 = \mathcal{O}(1)$

   Therefore, $\exists n_0, c_1 > 0 : \forall n > n_0$ such that
   $0 \le n + \frac{1}{n} = f(n) + g(n) \le c_1 f(n) \cdot g(n) = c_1$
   but this is a contradiction because $\forall n_0\ \exists n > n_0 : c_1 < n + \frac{1}{n}$ but we get that $n + \frac{1}{n} \le c_1$
   Thus, the Claim is incorrect ■
3. The claim is False.
   We will show such with a counter example: $f(n) = \frac{1}{n}$, $g(n) = n^2$
   let there be some $c > 0$ such that for $n_0 = min\{1, c\}$: $\forall n > n_0$ the following happens:
   $c \cdot f(n) = c \cdot \frac{1}{n} \le c \cdot \frac{1}{n_0} \le c \cdot 1 = c \le n^2$
   Therefore, $g(n) = \omega(f(n))$ but we will show that $f(n) + g(n) \ne \mathcal{O}(f(n))$ and from a theorem we will also get that $f(n) + g(n) \ne \Theta(f(n))$

   $\forall n_1 \exists n > n_1 : f(n) + g(n) > c \cdot f(n)$
   $\frac{1}{n} + n^2 > c \cdot \frac{1}{n}\ / \cdot n \implies 1 + n^3 > c \implies n^3 > c - 1 \implies n > \sqrt[3]{c-1}$

   We will choose $n = max\{n_1, \sqrt[3]{c-1}\}$ such that we get $f(n) + g(n) \ne \mathcal{O}f(n)$
   Therefore, according to the theorem mentioned before, $f(n) + g(n) \ne \Theta(f(n))$ ■

4. The claim is True.
It is given that $\forall c > 0\ \exists n_0 > 0 : \forall n > n_0$ such that: $0 \leq g(n) < c \cdot f(n)$

We would like to show that there
$\exists n_1, c_1, c_2 > 0 : \forall n > n_1 : 0 \leq f(n) + g(n) \leq f(n) + c \cdot f(n) = (c+1) \cdot f(n)$
Let $c_2 = c + 1$ such that we get by definition that $f(n) + g(n) = \mathcal{O}(f(n))$
Furthermore, Lets choose $c_1 = 1$ such that we get $0 \leq 1 \cdot f(n) = f(n) \leq f(n) + g(n)$
and we know that $f(n), g(n) \geq 0$ from the given.

Therefore, $f(n) + g(n) = \Omega(f(n))$
Now, according to a theorem since we found that $f(n) + g(n) = \mathcal{O}(f(n))$ and
$f(n) + g(n) = \Omega(f(n)) \implies f(n) + g(n) = \Theta(f(n))$ ■
5. The claim is True.

Let $m = log(n)$ Therefore, n^{ϵ} using log theorems is the same as $2^{\epsilon \cdot m} = (2^{\epsilon})^m$
From the given rule $\forall a \in \mathbb{R}_{>}1$ and $d \in \mathbb{R}$ the following is true: $n^d = o(a^n)$
*$(log(n))^b \rightarrow m^b$
Using this rule, we get $m^b = o((2^{\epsilon})^m) \implies f(n) = o(n^{\epsilon}) = o(g(n))$ Thus proving the desired. ■
6. The claim is True.

We would like to prove the following: $\forall c > 0 \exists n_0 > 0 L \forall n > n_0 : 0 \leq c \cdot n! < 2^{nlogn}$
Let there be some $c > 0$. We will choose $n_0 = c : \forall n > n_0$ the following occurs:
$2^{nlogn} = 2^{logn^n} > c \cdot n! > c \cdot n_0! \geq 0$
(The third move if true because $n \cdot n \cdot \ldots n > n \cdot (n-1) \cdot \ldots \cdot 1$ and with log rules,
$2^{log(x)} = x$)
Therefore, by definition we get that $f(n) = \omega(n!)$ ■
7. The claim is True.

For $k > 2$ where $k \in \mathbb{R}$, $f(n) = s^{\frac{n}{k} \cdot logn} = \sqrt[k]{n^n}$ According to log rules.

We want to prove that $\forall c \exists n > n_0 : 0 < \sqrt[k]{n^n} < c \cdot n!$ which implies that $f(n) = o(n!)$

Let there be some arbitrary $c > 0$ that for some $n_0 = max\{1, \frac{1}{c}\}, \forall n > n_0, n \in \mathbb{N}$ such that

$\sqrt[k]{n^n} = n^{\frac{n}{k}} \leq n^{\frac{n}{2}}$

now we will show that $n^{\frac{n}{2}} \leq n!$ which is the same as proving $n^n < (n!)^2$

$(n!)^2 = (1 \cdot n)(2 \cdot (n-1))(3 \cdot (n-2)) \cdot \ldots \cdot ((n-2) \cdot 3)((n-1) \cdot 2)(n \cdot 1) =$
$n \cdot (2n-2) \cdot (3n-6) \cdot \ldots \cdot (3n-6)(2n-2) \cdot n = \prod_{k=1}^{n} k \cdot (n-(k-1))$

we will show that $k(n-(k-1)) > n$ for all $1 < k < n$ since when $k = 1, n$ we get that the expression equals n. We will open up the expression:

$k(n+1-k)-n>0 \implies k\cdot n+k-k^2-n>0 \implies n\cdot(k-1)-k\cdot(k-1)>0$
$\implies (k-1)\cdot(n-k)>0 \implies$ the expression is positive $\forall n>2, n>k>1$
Thus we get
$1\cdot n=n, 2\cdot(n-1)>n, 3\cdot(n-2)>n, \ldots, (n-2)\cdot 3, 2\cdot(n-1)>n, n\cdot 1=n$
$\implies$ if we multiply each of these expressions we end up getting $(n!)^2>n^n$ thus proving what we wanted.
Which is a total length of n separate multiplications where each multiplication is greater or equal to n.
Therefore, we get the desired and we showed that $(n!)^2>n^n \implies n!>n^{\frac{n}{2}}$

Therefore, we get that $\sqrt[k]{n^n}=n^{\frac{n}{k}}\leq n^{\frac{n}{2}}<n!\leq n_0\cdot n!<c\cdot n! \implies f(n)=o(n!)$ ■

Question 2

1. The claim is True.

   We know that there exists $c>0$ that for every $n\in\mathbb{Z}_{>}0$ exists $I, |I|=n$ such that $T_A(I)\geq cg(n)$. We also know that $T_A(n)=max_{I:|I|=n}T_A(I)$. For every instance $I, |I|=n$ in $\{I_n\}$ we get that $0\leq cg(n)\leq T_A(I)\leq T_A(n)$.

   Therefore, by the definition there exists (The same c, n_0 that we got from the given) $c, n_0>0$ that for every $n>n_0$, $0\leq cg(n)\leq T_A(n)$. Thus, $T_A(n)=\Omega(g(n))$.
2. The claim is False.

   Function A is a counter example:
   n = A.length
   for i to n do {
   i = 2 * i
   }

   For this algorithm, the time complexity is $\Theta(\log n)$, so by the definition there exists $c_1, c_2, n_0>0$ such that $\forall n>n_0: 0\leq c_1\cdot logn\leq T_A(n)\leq c_2\cdot\log n$

   so we can see that $T_A(I)\leq T_A(n)\leq c_2\cdot\log n\leq c\cdot n^2 \implies T_A(I)\leq c\cdot n^2$
   for every $n>n_0$ but $T_A(n)\neq\Omega(n)$ like we asked since the upper bound is $logn$. ■
3. The claim is True.
   If there exists $c, n_0>0$ that for all $n\geq n_0$ and all the instance $I, |I|=n$ there $T_A(I)<c\cdot g(n)\leq c\cdot g(n)$ so by the definition:
   $T_A(I)=O(g(n)) \implies T_A(n)=max_{I:|I|=n}T_A(I)=O(g(n))$.
4. The claim is False.
   A counter example:
   n = A.length
   If n % 2 = 0{
   for i = 0 to n^3 do{
   print(“hi”)

```
}}
else{
print("by")
}
```

we can see that for every $n_0 > 0$ there will be an instance where n is odd, where $n \geq n_0$ such that $T_A(I) \leq c \cdot n^2$ since the runtime is $\Theta(1)$ shown by the last line in the function. But there is an instance I that $\forall\ n \geq n_0 : T_A(I) = \Theta(n^3)$ which can be seen in the for loop which runs from i to n^3. Therefore $T_A(n) = max_{I:|I|=n} T_A(I) \neq O(n^2)$.

Question 3:

Line 1 takes $\Theta(1)$ time.

Every iteration of the inner loop in lines 7-9 takes $\Theta(1)$ time.

In one iteration of the inner loop in lines 4-9, the most inner loop it runs at a worst case of $\log(n)$ iteration ($2^k = n \rightarrow k = \log(n)$). Therefore, in every iteration of the most inner loop (line 7) we get $(\log(n) \cdot \Theta(1) =)\ \Theta(\log(n))$ runtime.

In one iteration of the for outer loop in lines 2-9, the first inner loop run at the worst case $\log(n)$ iteration ($2^j = n \rightarrow j = \log(n)$). Therefore, in every iteration of the for loop we get $(\log(n) \cdot \Theta(\log(n)) =)\ \Theta((\log(n))^2)$ time.

The for loop runs $\lfloor \frac{n}{2} \rfloor$ to $n = \Theta(n)$ time, so the time complexity of the algorithm overall is the multiplication of each inner loop which results in: $(\Theta(n) \cdot \Theta((\log(n))^2) =) \Theta(n((\log(n))^2))$. $\blacksquare$

Question 4:

Firstly we will analyze the function called $FindPow(n)$
we start on lines 1,2 by instantiating k, p to $k = 1, p = 0$ which takes $\Theta(1)$ time each.
given n as an input we have a $while$ loop on line 3 that runs from $k = 1$ to n where the condition for when $k = n$ occurs according to line 4 when $2^k = n \implies k = logn$

p is equal to the amount of iterations that the $while$ loop runs which we found to be $logn$ thus $FindPow(n)$ will return the value of $p - 1$ which is $logn$.

calculating asymptotic bound of $Alg_4(n)$
on line 1 we call the $FindPow$ function and give it the value n. Therefore, $p := logn$ and so far the function will run $\Theta(logn)$ which is equal to the asymptotic runtime of $FindPow(n)$.

all the lines that are relevant to $binary$ variable run $\Theta(1)$ time and therefore since we found that the minimal runtime so far is $logn$ thus it won't affect the asymptotic runtime of the function.

on line 3, $n = n - 2^p = n - 2^{logn} = n - n = 0 \implies n = 0$

on line 4 we have a $loop$ that runs from $p-1$ to 0 meaning that it runs $logn$ times.
we have a variable called $NewP$ on line 5 that is defined to be $FindPow(n)$ when $n=0$ and n doesn't get changed since the condition on line 6 is never true until $i=0$ which only occurs on the last iteration of the loop. Therefore, until then n remains 0. In the last iteration when it's True then the function will run lines 7, 8 which are anyways have $\Theta(1)$ runtime. In all the other iterations as the first condition is false we will enter the $else$ condition on line 9, 10 which will run a total of $\Theta(1)$ time.

In addition line 11 which just returns the binary variable will run $\Theta(1)$ time.

In total we get that the functions runs:
$\Theta(log(n)) + 2 \cdot \Theta(1) + \Theta(log(n)) \cdot \Theta(1) \implies$ runtime = $\Theta(log(n)) \implies \Omega(logn)$ and $\mathcal{O}(logn)$ ■

Question 5:

We will analyze the runtime of $Alg_5(n)$.

Lines 1 run $\Theta(1)$ and instantiates the following variables with the value of 0: x, y, k

Line 2 is a loop that runs from 1 to $2n$ thus runs at $\Theta(n)$.

The first inner loop on line 3 runs a total of i^2 throughout each iteration of i
The second inner loop starts at a value of $t = k := i^2$
Lines 7 through 11 are negligible as they run $\Theta(1)$ each. Therefore, as the runtime will be greater than $\Theta(1)$ we can ignore their runtime relative to the total function runtime.

The second inner loop ($while$) in each i iteration will run $2 \cdot i^2$ time since line 12 tells us that $t- = 0.5$ in each of the inner loops iteration.

We have two equations here that their runtime is in parallel.

Therefore the total runtime is equal to the following:
$\sum_{i=1}^{2n} i^2 + \sum_{i=1}^{2n} (2 \cdot i^2) = \frac{1}{3} \cdot (n \cdot (8 \cdot n^2 + 6 \cdot n + 7))$ + $\frac{2}{3} \cdot (n \cdot (8 \cdot n^2 + 6 \cdot n + 7)) = \Theta(n^3) + \Theta(n^3) = \Theta(n^3)$

Total runtime = $\Theta(n^3)$ ■