<u>Data Structures & Algorithms – Problem set 1</u>

Question 1:

Base step: n=1 → $4^1 + 15 \cdot 1 - 1 = 18$, which is dividable by 9.

Inductive step: Let's assume $4^n + 15n - 1$ is true for n=k, Hence $4^k + 15k - 1$ is true

Let's prove for n=k+1:

- ⇨ $4^{k+1} + 15(k+1) - 1$ => $4x4^k + 15k + 15 - 1$ => $\left(4^k + 15k - 1\right) + \ (3x4^k + 15)$
- ⇨ The first part assumed to be dividable by 9, let's prove by second induction the second part $(3x4^k + 15)$ is also dividable by 9.
- ⇨ Base step: n=1 → $3x4^1 + 15 = 27$, which is dividable by 9.
- ⇨ Inductive step: Let's assume $3x4^n + 15$ is true for n=k, Hence $3x4^k + 15$ is true
- ⇨ Let's prove for n=k+1:
- ⇨ $3x4^{k+1} + 15 \ => 12x4^k + 15 = > 9x4^k + (3x4^k + 15)$, both part are dividable by 9.
- ⇨ Hence, $4^n + 15n - 1$ is dividable by 9.

Question 2:

Let's prove: $C_1 n^5 \geq \frac{n^5}{3} - 10n^3 + 7n - 25 \geq C_2 n^5$.

Let's solve left side first:

$n^5 \geq \frac{n^5}{3} \geq \frac{n^5}{3} - 10n^3 + 7n - 25$, for all n>0, hence, $C_1 = 1$.

Now right side:

$\frac{n^5}{3} - 10n^3 + 7n - 25 \geq \frac{n^5}{3} - 10n^3 - 25 \geq \frac{n^5}{3} - \frac{n^5}{12} - \frac{n^5}{12} \geq \frac{n^5}{6}$, true when $\frac{n^5}{12} \geq 10n^3 and\ \frac{n^5}{12} \geq 25$

$=> \ n \geq 11.$

Hence C_2=1/6 for and $C_1 = 1$ for all $n_0 \geq 11$

Question 3:

The order of the functions (ascending order) :

1. F2 = $n^{\frac{2}{\log_2 n}} = 2^{logn\frac{2}{\log_2 n}} = 2^2$ = O(1)
2. F7 = $4^{2021} + 2020 = \ O(1)$
3. F5 = $\log_2 n^{10} = 10 \log_2 n = \ O(log\, n)$

4. F4 = n/10 = $O(n)$
5. $\text{F1} = \log_2 2^n n^{10} = \log_2 2^n + \log_2 n^{10} = n + 10\log_2 n = O(n)$
6. $\text{F6} = 3^{\log_2 n} = 2^{\log_2 3^{\log_2 n}} = 2^{\log_2 n * \log_2 3} = n^{\log_2 3}$ = O(n^2) for ∀n≥ 0 $and\ C = 1$
7. F3 = $2^{3n} = 8^n = O(8^n)$

Proof for the pairs:

$\underline{\boldsymbol{F2 = \theta(F7)}}$:

$\lim\limits_{n\to\infty} \frac{2^2}{4^{2021}+2020}$ which is a constant.

$\underline{\boldsymbol{F4 = \theta(F1)}}$:

$\lim\limits_{n\to\infty} \frac{10\log(n)+n}{n/10} = \frac{100\log(n)+10n}{n} = \frac{100\log(n)}{n} + 10 = 10$ which is a constant.

1. $\underline{\boldsymbol{f7 = O(f5)}}$**:**

   For every n$\geq$ 2 and C_1=$4^{2021} + 2020$, F7=$4^{2021} + 2020 \leq (4^{2021} + 2020)\log(n)$ = C_1*F5

2. **F5 = O(F4):**
   Proof by induction:
   Claim: For all n≥1, log(n)≤n.
   Base: for n=1, since 0<1.
   Now suppose n≥1 and log(n)≤n. Then we need to prove log(n+1)≤n+1. Because n≥1 we can assume that log(n+1)≤log(2n) = log(n)+1≤n+1 by the inductive hypothesis.

3. **F4 = 0(F6):**
   Because $\log_2 3 > 1$ for c$\geq$ 1 for every n$\geq$ 0: n/10 ≤ n ≤ c*$n^{\log_2 3}$

4. **F3 = 0(F6):**

   $\log(f3)$ = $\log_2 3 * \log(n) \leq \log(n) \leq 3n \leq \log(f6)$ for any n$\geq$ 0 as log(n) is monotone, thus F3$\leq$ $F6\ for\ any\ n \geq 0$ and c $= 1$

Question 4:

The definition of Big O Notation is based on the fact that there exist two **<u>constants</u>** C, n_0 that holds:

$\leq f(n) \leq C_2 g(n)$ for any N>n_0. These two cannot be changed and should hold for every value.

In the induction attempt in the question the assumption claimed that there is a constant that holds the proof. But in the proof step in order to verify the assumption C was changed which undermine the whole proof. C was needed to be changed in order for the proof to hold which directly contradicts the definition the induction and bigO.

Question 5:

A.

$f(n) = \boldsymbol{\theta}(g(n))$

This means, $g(n) \leq O(f(n))\ and\ fn \leq O(g(n))$

let's prove: $\log(g(n)) \leq O(\log(f(n)))\ and\ \log(f(n)) \leq O(\log(g(n)))$

Let's solve the right side:

$\log(f(n)) \leq \log(C_2 g(n)) \leq \log(C_2) + \log(g(n))$

If $C_2 > 1\ then$:

$\log(f(n)) \leq \log(C_2) + \log(g(n)) \leq \log(C_2) + \log(C_2)\log(g(n)) \leq (1 + \log(C_2))\ \log(g(n))$

$C = (1 + \log(C_2))\ for\ any\ n \geq 2$

If $C_2 \leq 1\ then$:

$\log(f(n)) \leq \log(C_2) + \log(g(n)) \leq \log(g(n))$

$C = 1\ for\ any\ n \geq 2$

Thus $\log(f(n)) \leq O(\log(g(n)))$

For the left side:

Reversing the role of F(n) and G(n) also show that $\log(g(n)) \leq O(\log(f(n)))$

which enough to prove that log(f(n)) = Θ(log(g(n)))

Thus the statement is true.

B.

Let for example f(n)=2n and g(n)=n => $2^{f(n)} = 4^n\ and\ 2^{g(n)} = 2^n$.

Let's test it with the limit definition of θ:

$\lim_{n\to\infty} \frac{4^n}{2^n} = \lim_{n\to\infty} 2^n = \infty$, thus the statement is false.

Question 6:

```
Function findPair(L,k): {
    NodeA = L.First
    For I= 1 to length L -1:
        For J=i+1 to length L:
            NodeB = NodeA.next
            If NodeA.value + NodeB.value = K return True.
             NodeB = NodeB.next
        NodeA =NodeA.next
    Return False
}
```

In the best case scenario the first two items will justify the condition which will account as 1 comparison = O(1). In the worst case scenario none of pair items will validate the requirements, in this case the program will run through the whole list and for each item on the list it will ran again on the rest of the list to try to find an item that together will justify the condition, thus it will run n-1 of the first run, then n-2, n-3, till 1 -> n-1 times. The sum of this arithmetic sequence is $\frac{n(n-1)}{2}$ = O(n^2)