Topic: Time complexity

1. **Time Complexity (40 points)**
   a. **Sort** the following functions in ascending order of their growth. For example- $n^2<n^3$, etc.
   2^n, $\log n$, $\log\log n$, n^2, n, $\sqrt{n}$, $n!$, n^3, $n^{3/2}$, $n \log n$, e^{n+1}, $n^2 \log n$ **5**
   Hint: you can plot the functions in any graphing tool like Desmos to visualize their growth.
   Note: Just writing the order should suffice. Visualise/reason is only for your own understanding.
   b. **Prove** the following (you can use any formal induction/other theoretical method): **2.5x6**
      i. $n^2 + 15n - 3 = \theta(n^2)$
      ii. $4n^3 - 7n^2 + 15n - 3 = \theta(n^3)$
      iii. $T(n) = 4T(n/2) + n = \theta(n^2)$
      iv. $T(n) = 2T(n/2) + n^3 = \theta(n^3)$
      v. $T(n) = T(n/4) + T(5n/8) + n = O(n)$
      vi. $T(n) = T(n/3) + T(4n/9) + n = O(n)$
   c. **Let, the time complexity of each of the following code snippets be T(n). Find out a tight bound for T(n) in Big-Theta (θ) notation.** **5+5**

```
1. count = 0;
   for (i=1, i<=n; i*=2)
      for (j=1, j<=i; j++)
        count++;
2. p = 3
   while (p < n)
      p = p * p
```

   d. For the following code- **5+5**
      i. **Derive** the recurrence function: T(n)
      ii. **Find** its time complexity.

```
int ternary_search(int l,int r, int x)
{
    if(r>=l)
    {
        int mid1 = l + (r-l)/3;
        int mid2 = r -  (r-l)/3;
        if(ar[mid1] == x)
            return mid1;
        if(ar[mid2] == x)
            return mid2;
        if(x<ar[mid1])
            return ternary_search(l,mid1-1,x);
        else if(x>ar[mid2])
            return ternary_search(mid2+1,r,x);
```

Topic: searching (Linear , Binary)

1. **Searching (10 + 5 = 15 points)**
    - **a.** You are given two arrays: Arr1 and Arr2.

      Arr1 will be given sorted. For each element v in Arr2, you need to **write a code/pseudo code** that will print the number of elements in Arr1 that is **less than or equal** to v. For example: if I give you two arrays of size 5 and 4

      5 4 [size of two arrays]

      Arr1 = 1 3 5 7 9

      Arr2 = 6 4 8

      The output should be: 3 2 4

      Firstly, you should search how many numbers are there in Arr1 which are less than 6. There are 1,3,5 which are less than 6 (total 3 numbers). So the answer for 6 will be 3.

      After that, you will do the same thing for 4 and 8 and output the corresponding answers which are 2 and 4. **Your searching method should not take more than *O(log n)* time.**

Sample input	Sample output

5 5
1 1 2 2 5
3 1 4 1 5
	4 2 4 2 5

    - b. **Show** the calculation of the time complexity for your written code.