1(a) Formally prove that the following function is $\Theta(n^4)$.

$$f(n) = 2n^4 + 3n^3 - 200n^2 - 2000n$$

1(b) Formally prove that a polynomial of degree $k$ (where $k$ is a constant) is $O(n^k)$. You can assume the coefficients $a_k, a_{k-1}, \cdots, a_0$ are all positive.

$$P_k(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0$$
2(a) Find the exact solution of the following recurrence by using the repeated substitution method. Assume $n$ is a power of 2.

$$T(n) = \begin{cases} 
2T(n/2) + 1, & n \geq 2 \\
0, & n = 1. 
\end{cases}$$

2(b) Prove by induction that the following recurrence has the solution $f(n) = A4^n + B$ and find the constants $A, B$.

$$f(n) = \begin{cases} 
4f(n-1) + 1, & n \geq 1 \\
0, & n = 0. 
\end{cases}$$
3. Given a sorted array $A[0 \ldots n - 1]$. We want an algorithm that searches for a given element $X$ and returns the largest index $i$ where $A[i] = X$ is found, and returns $-1$ if $X$ is not found. Write a modified version of binary search function (using C or C++) to do this task.
4. Consider the task of computing the **height** of a binary tree. (We define the height of a leaf node to be 0.) Suppose each node contains pointers LCHILD (left-child) and RCHILD (right-child).

(a) The following function is an attempt in doing this job but is very inefficient. Explain very briefly why it is inefficient.

```c
int height(ptr r)
{
    if (r==NULL) return -1;
    if (height(r->LCHILD) > height(r->RCHILD))
        return (1+height(r->LCHILD));
    else return (1+height(r->RCHILD));
}
```

(b) Let $T(n)$ be the worst-case running time of the above algorithm for computing the height of a tree with $n$ nodes. Write a recurrence for $T(n)$ in terms of: $n_L =$ the number of nodes in the left subtree, and $n_R =$ the number of nodes in the right subtree. (Suppose that the height of the left subtree is $> \text{height of the right subtree}$.)

(c) Rewrite the above algorithm, $height(r)$, to correct the inefficiency.
5(a) Insert element 7 in the following heap.
(First complete the picture by drawing lines from each node to its children.)

```
  8
 /   \
15   12
 /   /   \
20 15 12 13
 /   /   /   \
23 21 15 17 13 15 14 13
 /     /     /     \
25 24 22 21 15 16 17 18 15 14 15 16 15
```

What is the time complexity of one insert operation for a heap of $n$ elements? Explain briefly.

5(b) Do a Delete-Min operation on the following heap.

```
  8
 /   \
15   12
 /   /   \
20 15 12 13
 /   /   /   \
23 21 15 17 13 15 14 13
 /     /     /     \
25 24 22 21 15 16 17 18 15 14 15 16 15
```

What is the time complexity of one Delete-Min operation for a heap of $n$ elements? Explain briefly.