Leetcode

Leetcode 1 Solution:-

```
class Solution {
    public String longestWord(String[] words) {
        Arrays.sort(words, (a, b) -> a.length() - b.length()); // sort
words by non-decreasing length
        HashMap<String, Boolean> goodWords = new HashMap(); // lookup for
good words
        int maxLength = 0;
        String ans = "";
        for (String word : words) {
            if (word.length() == 1) {
                goodWords.put(word, true);
            } else if (goodWords.containsKey(word.substring(0,
word.length() - 1))) {
                // word with length - 1 prefix is good
                goodWords.put(word, true);
            }
            if (goodWords.containsKey(word)) {
                if (maxLength < word.length()) { // find longer word
                    maxLength = word.length();
                    ans = word;
                } else if (maxLength == word.length()
                        && ans.compareTo(word) > 0) { // find
lexicographically smaller word
                    ans = word;
                }
            }
        }
        return ans;
    }
}
```

Leetcode 2 Solution:-

```
class Solution {
    public TreeNode insertIntoBST(TreeNode root, int val) {
        if(root == null) return new TreeNode(val);
        TreeNode curr = root;
        while(true){
            if(curr.val >= val){
                if(curr.left != null) curr = curr.left;
                else {
                    curr.left = new TreeNode(val);
                    break;
                }
            }
            else{
                if(curr.right != null) curr = curr.right;
                else {
                    curr.right = new TreeNode(val);
                    break;
                }
            }

        }
        return root;
    }
}
```