Please use this Google doc to code during your interview. To free your hands for coding, we recommend that you use a headset or a phone with speaker option.

1. Given a sorted list of strings; build a tree, where each string is represented by one node, and each node A is an ancestor of another node B iff A's word is a prefix of B's word.

```
{"a", "ab", "app", "apple", "b", "banana"}

              ""

      a             b
    ab   app            banana
         apple



        ""
a app apple b banana



        ""
    a
ab

class TreeNode {
 String val;
 ArrayList<TreeNode> children;

 public TreeNode (String val) {
  this.val = val;
  }
}

public TreeNode buildTree(List<String> input) {
 TreeNode root = new TreeNode("");

 for (int i = 0; i< input.size(); i++) {
   helper(root,input.get(i));
 }
 return root;
}

//check recursively if curStr can be inserted as a
child in subtree of root,
// if not , insert curStr as a direct child of root
public boolean helper(TreeNode root, String curStr) {
  boolean canInserInSub = false;
  if (curStr.contains(root.val)) {
   // root str is a prefix of curStr
      for (TreeNode child: root.children) {
       if (helper(child, curStr)) {
        // we can insert curStr under a child of root
        canInsertInSub = true;
        break;
       } //if
      } // for

      if (!canInserInSub) {
      // insert curStr as a direct child of root
       root.children.add(new TreeNode(curStr));
      }
  }// if (curStr.contains(root.val))
   else {  // root.val is not a prefix of curStr
    return false;
  }
  return true;
}
```