1、以下函数用于将一颗二叉搜索树转换成一个有序的双向链表。要求不能创建任何新的节点，只能调整树种节点指针的指向。

如输入下图中左边的二叉搜索树，则输出转换后的排序双向链表：

      10

    /      \

   6      14

  /  \      /  \

4   8  12  16

转换成：

 4 <=> 6 <=> 8 <=> 10 <=> 12  <=> 14 <=> 16

请指出程序代码中错误的地方（问题不止一处，请尽量找出所有你认为错误的地方）：

1  #include <stack>

2  using namespace std;

3

4  struct TreeNode {

5        int val;

6        TreeNode \*left, \*right;

7  };

8

9  TreeNode\* Convert(TreeNode\* root){

10         if (root == NULL)

11             return root;

12

13         TreeNode\* listHead = NULL;

14         TreeNode\* listLastNode = NULL;

15

16         stack<TreeNode\*> s;

17         while(root){

18             while(root){

19                 root=root->left;

20                 s.push(root);

21             }

22             root=s.top();

23             s.pop();

24             if (listHead == NULL){

25                 listHead = root;

26             }else{

27                 listLastNode->right = root;

28             }

29             listLastNode = root;

30             root= root->right;

31         }

32         return listHead;

33 }

2、以下代码为适用于ListView的Adapter的部分代码，请根据代码解释下

1）getView方法中对convertView是否为空做了不同处理，目的是什么？

2）viewholder有什么作用？

private List<String> mData = new ArrayList<>();

…

private static class ViewHolder {

   TextView text;

   public ViewHolder(View rootView) {

       text = (TextView)rootView.findViewById(R.id.text);

   }

   public void bind(String item) {

       ...

   }

}

@Override

public View getView(int position, View convertView, ViewGroup parent) {

   String item = mData.get(position);

   ViewHolder viewHolder;

   if (null == convertView) {

       convertView = LayoutInflater.from(mContext).inflate(R.layout.list\_item, parent, false);

       viewHolder = new ViewHolder(convertView);

       convertView.setTag(viewHolder);

   } else {

       viewHolder = (ViewHolder) convertView.getTag();

   }

   viewHolder.bind(item);

   return convertView;

}

3、列表卡顿怎么优化？首先卡顿怎么量化；其次怎么发现造成卡顿的原因；针对可能发现的问题，又如何解决？请设计一套方案。

4、有三只球队，每只球队编号分别为球队1，球队2，球队3，这三只球队一共需要进行 n 场比赛。现在已经踢完了k场比赛，每场比赛不能打平，踢赢一场比赛得一分，输了不得分不减分。已知球队1和球队2的比分相差d1分，球队2和球队3的比分相差d2分，每场比赛可以任意选择两只队伍进行。求如果打完最后的 (n-k) 场比赛，有没有可能三只球队的分数打平。   

**输入描述:**

第一行包含一个数字 t (1 <= t <= 10)  
接下来的t行每行包括四个数字 n, k, d1, d2(1 <= n <= 10^12; 0 <= k <= n, 0 <= d1, d2 <= k)

**输出描述:**

每行的比分数据，最终三只球队若能够打平，则输出“yes”，否则输出“no”

**输入例子1:**

2

3 3 0 0

3 3 3 3

**输出例子1:**

yes

no

**例子说明1:**

case1: 球队1和球队2 差0分，球队2 和球队3也差0分，所以可能的赛得分是三只球队各得1分  
case2: 球队1和球队2差3分，球队2和球队3差3分，所以可能的得分是 球队1得0分，球队2得3分, 球队3 得0分，比赛已经全部结束因此最终不能打平。

5、有一个仅包含’a’和’b’两种字符的字符串s，长度为n，每次操作可以把一个字符做一次转换（把一个’a’设置为’b’，或者把一个’b’置成’a’)；但是操作的次数有上限m，问在有限的操作数范围内，能够得到最大连续的相同字符的子串的长度是多少。 

**输入描述:**

第一行两个整数 n , m (1<=m<=n<=50000)，第二行为长度为n且只包含’a’和’b’的字符串s。

**输出描述:**

输出在操作次数不超过 m 的情况下，能够得到的 最大连续 全’a’子串或全’b’子串的长度。

**输入例子1:**

8 1

aabaabaa

**输出例子1:**

5

**例子说明1:**

把第一个 'b' 或者第二个 'b' 置成 'a'，可得到长度为 5 的全 'a' 子串。