题目描述：

根据给定的二叉树结构描述字符串，输出该二叉树按照中序遍历结果字符串。中序遍历顺序为：左子树，根结点，右子树。

输入描述：

由大小写字母、左右大括号、逗号组成的字符串：

1、字母代表一个节点值，左右括号内包含该节点的子节点。

2、左右子节点使用逗号分隔，逗号前为空则表示左子节点为空，没有逗号则表示右子节点为空。

3、二叉树节点数最大不超过 100。

注：输入字符串格式是正确的，无需考虑格式错误的情况。

输出描述：

输出一个字符串，为二叉树中序遍历各节点值的拼接结果。

补充说明：

中序遍历是二叉树遍历的一种，遍历方式是首先遍历左子树，然后访问根结点，最后遍历右子树。

示例 1

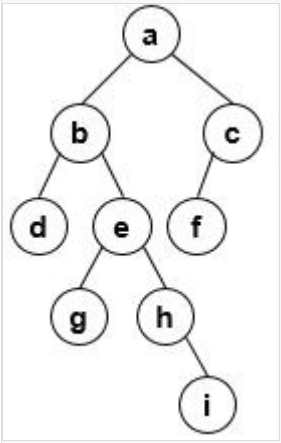
输入：

```
a{b{d,e{g,h{,i}}},c{f}}
```

输出：

```
dbgehiafc
```

说明：



中序遍历，首先遍历左子树，再访问根节点，最后遍历右子树，比如：

a 有左子树，访问其左子树

b 有左子树，访问其左子树

d 没有左子树，读取值"d"

b 的左子树已经访问，读取值‘‘b’’，再访问其右子树

e 有左子树，访问其左子树

g 没有左子树，读取其值‘‘g’’

e 的左子树已经访问，读取值‘‘e’’，再访问其右子树

依次类推……

```
import sys

line = sys.stdin.readline().strip()

class Node:
    def __init__(self, val, left=None, right=None, parent=None):
        self.val = val
        self.left = left
        self.right = right
        self.parent = parent

def solve(line):
    if len(line) == 1:
        return line
    root = Node(line[0])
    current = root
    temp = current
    flag = 'l'
    line = line[1:]
#     print(line)
    for c in line:
        if c == '{':
            current = temp
            flag = 'l'
        elif c == ',':
            flag = 'r'
        elif c == '}':
            current = current.parent
        else:
#             print(c, current.val, flag)
```

```
                    val = c
                    if flag == 'l':
                        current.left = Node(c, parent=current)
                        temp = current.left
                    else:
                        current.right = Node(c, parent=current)
                        temp = current.right

#         print(root.left.val)

        def search(node):
            if not node:
                return
            search(node.left)
            print(node.val, end='')
            search(node.right)
        search(root)

solve(line)
```