全面教育 101

Midterm 2复习题
解决方案

1. 从最低到最高的渐近增长率对以下函数进行排序。

1) 2
2) ln( ) 3)
4) 2 ln( )
5) ln(ln(n))
6) √ 7) 8)
ln(9) √

2^n

_ 2^n)

将你的答案写成集合 {1, 2, 3, 4, 5, 6, 7, 8, 9} 的排列,按照要求的顺序给出上述函数的相应行号（从左到右,增长最慢的函数到增长最快的功能。）不需要任何理由。

解:5 8 9 4 3 2 6 7 1

2. 考虑来自 pa5 但没有cleanup () 函数的列表 ADT 。编写一个 C++ 客户端函数
标题

void RemoveDuplicates(列表& L)

它与cleanup()做同样的事情,只是光标结束的位置无关紧要。换句话说,调用RemoveDuplicates(L)将改变列表 L,使其仅包含其每个数据项的第一次出现。为此,您可以使用 List.h 中除 cleanup()之外的所有 ADT 操作。

解决方案:

```
void RemoveDuplicates(List& L){

        诠释 p, x, y;

        L.moveFront(); p = 0;
        while( p<L.length() ){ x
        = L.moveNext();
                while(L.position()<L.length()){ y
                = L.moveNext();如果( y==x ){ L.eraseBefore();



                        }

                } p++;
                while(L.position()>p){ L.movePrev();

                }
        }
}
```

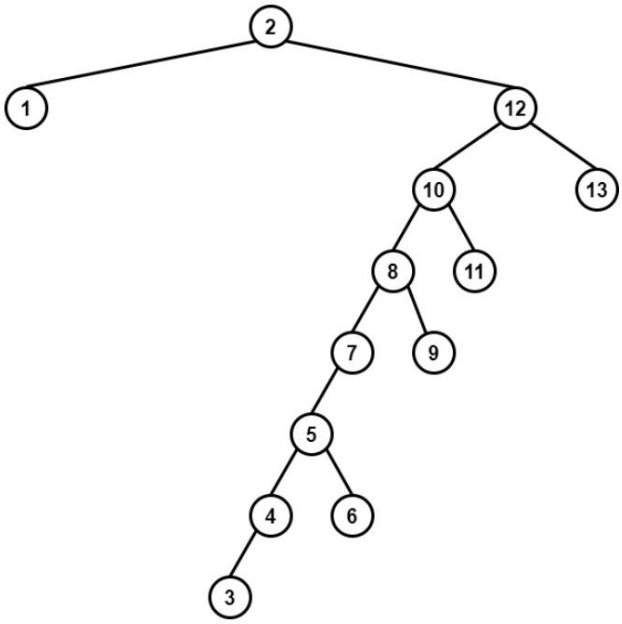
替代解决方案:

```
void RemoveDuplicates2(List& L){

        诠释 p, x;

        L.moveFront(); p = 0;
        while( p<L.length() ){ x
        = L.moveNext(); p = L.findNext(x);
                while(p>=0){ L.eraseBefore(); p
                = L.findNext(x);



                }
                L.moveFront(); p =
                L.findNext(x);
        }
}
```

3. 假设是一个包含键 {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13} 的二叉搜索树。假设前序树遍历按顺序打印键：2、1、12、10、8、7、5、4、3、6、9、11、13，而后序树遍历打印按键顺序：1、3、4、6、5、7、9、8、11、10、13、12、2。确定 的结构。

（注意：两次树遍历中只有一个是真正必要的，因为它们中的每一个都唯一地确定了 的结构。）通过绘制树的图片或通过构建一个表来给出每个节点的父节点来展示您的解决方案。

解决方案1（图片）：



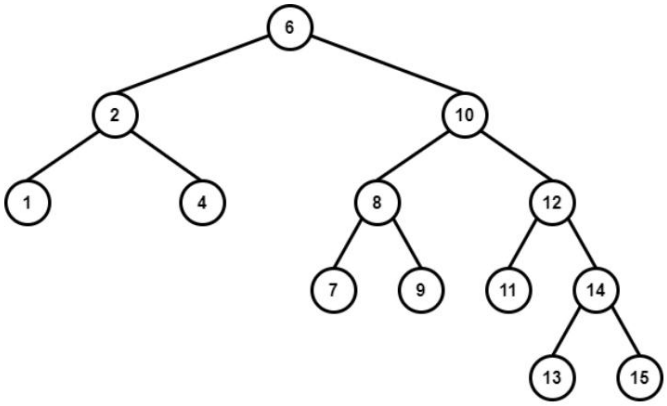
解决方案2（表）：

节点父节点 2	
1	
2	无
3	4
4	5 7
5	
6	5
7	8
8	10
9	8
10	12
11	10
12	2
13	12

4. 使用TreeInsert()算法插入以下键:6, 2, 1, 4, 10, 8, 7, 9, 12, 11, 14, 13, 15
（按顺序)进入一个最初为空的 BST。

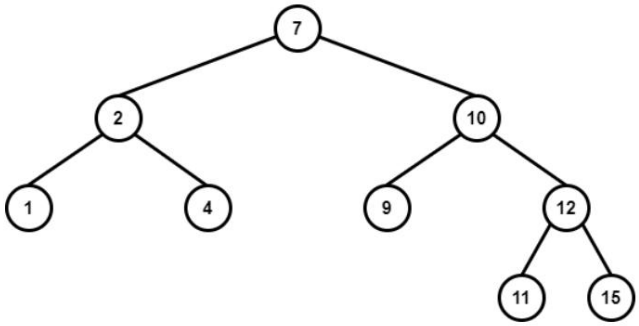
A。 （10分)画出结果BST

解决方案:



b. （10 分)使用Delete()算法从中删除以下键:8、6、13、14（按顺序）
你在部分（a)中绘制的 BST,然后绘制生成的树。

解决方案:



5. 假设我们通过执行以下操作从 pa5 更改列表 ADT

```
typedef 字符列表元素;
```

在 List.h 的开头,使其成为一个char列表而不是int。假设列表 L 完全由括号字符“(”和“)”组成。列表 L 称为格式正确的公式(WFF),前提条件是所有括号都可以成对匹配（打开和关闭）。例如“((()(()))”和“()()(())”是 WFF,而“((()()”和“((()))”不是。空列表被认为是一个 WFF。写一个带标题的客户端函数

```
bool isWFF(列表L)
```

根据L是否为 WFF ,返回真或假。 （提示:搜索相邻的匹配对并删除它们。如果L变为空,则返回 true。）

解决方案:

```
bool isWFF(列表L){

    诠释;

    // 删除匹配对 L.moveFront();
    while( L.length()>0 ){


        p = L.findNext(  )  );

        // p==-1 当且仅当未找到  )  时。 p==1 if and only if //  )  was found, but has no matching  (  on its left. 在
        这两种情况下,我们都会中断,因为不能删除任何匹配对。注意 // p==0从 findNext() 的规范来看是不可能的。if( p<2 ){ break;




        }

        // 删除匹配对  ()
        L.eraseBefore(); // 删除  )
        L.eraseBefore(); // 删除  (
    }

    返回 ( L.length()==0 );
}
```