1 remove 功能的实现

首先我们分为 detachMin 函数与 remove 函数两个部分:

1. 通过 detachMin 函数,我希望将被 remove 的节点通过指针替换成其右子树的最小节点。从而通过 detachMin 函数，我需要剥离该节点，并改变右子树的局部结构，将其剥离后的结构再连接起来，于是为了保证在不改变引用变量的前提下，改变树的局部结构，减少引入 parent 指针的操作，我采用了递归的方式找到右子树的最小节点，并将其赋值给 tem 后返回，再通过改变指向其指针为其指向其右节点的指针，成功实现了剥离节点与拼接。

2. 回到 remove 函数，root 仅有一个左（右）子节点以及无节点，root 为空的情况与示例代码相同。问题主要在于如何处理两个子节点的情况。首先我先调用 detachMin 函数，找到右子树的最小节点，并完善剥离节点处的结构后，用 oldNode 指向 root，后并改变了指向 root 指针指向 detachMin 返回的节点并改变了其左右节点为 root 的左右节点实现了替换，最终 delete oldNode，完成了对需要 remove 节点的内存回收。

2 测试结果及分析

我创建了 bst 树，并按照下列顺序插入 bst.insert (10); bst.insert (5); bst.insert (15); bst.insert (3); bst.insert (7); bst.insert (12); bst.insert (18); bst.insert (1);bst.inser (20)。下面为测试结果：

Initial Tree:
1
3
5
7
10
12
15
18
20
Minimum element: 1
Maximum element: 20
Contains 7? Yes
Contains 20? Yes
Tree after removing 7:（测试删除 leaf）
1
3
5
10
12
15
18
20
Tree after removing 15:（测试删除 two-children）
1
3

5
10
12
18
20
Tree after removing 10:（测试删除 root）
1
3
5
12
18
20
Tree after removing 3:（测试删除 one-child）
1
5
12
18
20
Tree after removing 2:（测试删除不存在节点）
1
5
12
18
20
Tree after making empty:
Empty tree
Tree after removing null root: 测试删除空树）
Empty tree
Is tree empty? Yes
如同结果和括号注释所示，测试了 remove 的主要功能以及边界情况，验证成功实现了 remove 的功能。