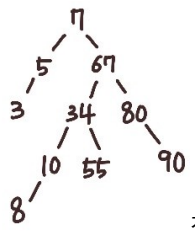
자료구조 레포트 #3

32181854 박준영

1. 수행결과

```
Microsoft Visual Studio 디버그 콘솔
3       5       7       8       10      34      55      67      80      90
3       5       7       8       34      67      80      90
2       3       5       7       8       9       20      34      43      67      80      90      100
2       3       5       7       8       34      43      67      80      90
```

2. 구현



첫번째 insert한 data들의 트리

0) main

Binary_Search_Tree

```cpp
#include <iostream>
#include "BST.h"
#include "BSTNode.h"
using namespace std;

int main() {
    BST t;
    //insert 10 keys
    t.insert(7);
    t.insert(67);
    t.insert(34);
    t.insert(80);
    t.insert(55);
    t.insert(10);
    t.insert(5);
    t.insert(8);
    t.insert(3);
    t.insert(90);

    t.inorder();
    cout << endl;

    if (t.search(10))
        t.remove(10);
    else cout << "there's no such key" << endl;
    if (t.search(55))
        t.remove(55);
    else cout << "there's no such key" << endl;

```

```
30      t.inorder();
31      cout << endl;
32
33      t.insert(20);
34      t.insert(2);
35      t.insert(9);
36      t.insert(43);
37      t.insert(100);
38
39      t.inorder();
40      cout << endl;
41
42      if (t.search(20)) t.remove(20);
43      else cout << "search: flase" << endl;
44      if (t.search(100)) t.remove(100);
45      else cout << "search: false" << endl;
46
47      t.inorder();
48      cout << endl;
49
50  }
```

insert -> search -> remove

1) BSTNode.h

Binary_Search_Tree

```
1   #pragma once
2   class BST;
3
4   class BSTNode
5   {
6   private:
7       BSTNode* leftchild;
8       BSTNode* rightchild;
9       int data;
10      friend BST;
11  public:
12      BSTNode(int);
13      ~BSTNode();
14  };
15
```

2) BSTNode.cpp

Binary_Search_Tree

```cpp
#include "BSTNode.h"
BSTNode::BSTNode(int x = 0) {
    leftchild = 0;
    rightchild = 0;
    data = x;
}
BSTNode::~BSTNode() { }
```

3) BST.h

Binary_Search_Tree

```cpp
#pragma once
#include "BSTNode.h"

class BST
{
private:
    BSTNode* root;
public:
    bool insert(int);
    bool remove(int);
    bool search(int);
    void inorder();
    void inorder(BSTNode*);
    BST();
    ~BST();
};
```

4) BST.cpp

Binary_Search_Tree → BST

```cpp
#include "BST.h"
#include <iostream>

//initiallization
BST::BST() {
    root = 0;
}
//insert new node
bool BST::insert(int x) {
    if (root == 0) {    //empty tree
        root = new BSTNode(x);
        return true;
    }
    BSTNode* p = root;
    BSTNode* q = 0;
    while (p) {
        q = p;
        if (p->data == x) return false;     //already exist
        else if (p->data > x) p = p->leftchild;
        else p = p->rightchild;
    }
    p = new BSTNode(x);
    if (q->data < x) q->rightchild = p;
    else q->leftchild = p;

    return true;
}
```

```
29  //delete node
30  bool BST::remove(int x) {
31      if (root == 0) return false;    //empty tree
32      BSTNode* p = root;
33      BSTNode* q = 0;
34
35      while (p) {
36          if (p->data == x) break;    //found node
37          q = p;
38          if (p->data > x) p = p->leftchild;
39          else p = p->rightchild;
40      }
41      if (!p) return false;   //there's no x
42
43      if (!(p->leftchild) && !(p->rightchild)) {  //leaf node
44          if (q->data > x) { q->leftchild = 0; }
45          else { q->rightchild = 0; }
46      }
47      else if (p->leftchild && p->rightchild) {   //two childs
48          BSTNode* t = p;
49          p = p->leftchild;
50          while (1) {
51              if (!p->rightchild) break;
52              q = p;
53              p = p->rightchild;
54          }
55          t->data = p->data;  //change node
56          q->rightchild = 0;
```

<remove 함수의 구현>

1. input data (=x) 탐색
2. leaf node, two childs, one childs 별로 다르게 삭제 수행

```
58      else {  //one child
59          if (q->data > x) q->leftchild = p->leftchild;
60          else q->rightchild = p->rightchild;
61      }
62      return true;
63  }
64  //search node
65  bool BST::search(int x) {
66      if (root == 0) return false;    //empty tree
67      BSTNode* p = root;
68      BSTNode* q = 0;
69      while (p) {
70          q = p;
71          if (p->data == x) break;    //found node
72          else if (p->data > x) p = p->leftchild;
73          else p = p->rightchild;
74      }
75      if (!p) return false;   //there's no x
76      else {
77          return true;
78      }
79  }
80  //inorder traversal
81  void BST::inorder() {
82      inorder(root);
83  }
84  void BST::inorder(BSTNode* CurrentNode) {
85      if (CurrentNode) {
86          inorder(CurrentNode->leftchild);
```

```
87          std::cout << CurrentNode->data << '₩t';
88          inorder(CurrentNode->rightchild);
89      }
90  }
91  BST::~BST() { }
```

inorder() 함수 중복

순환호출로 구현