HW10 실습: AVL 트리 (과제 아닙니다, 제출하지 않습니다.)

질문
pemds81718@gmail.com

1. 실습 내용

hw9.cpp 와 아래의 실행 예시를 참고하여 간단한 AVL Tree 를 구현합니다. 코드 자유롭게 수정 가능

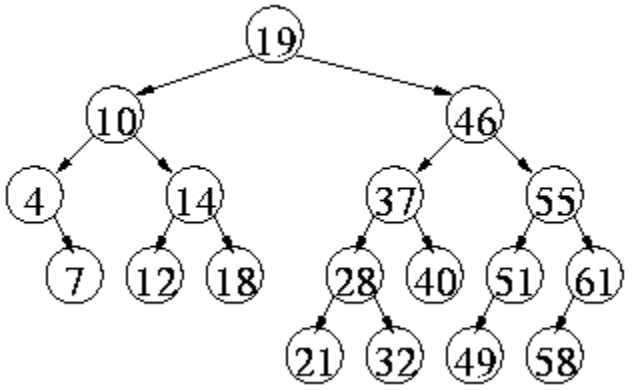
실행 예시:

```
[ta_hsc@localhost dsdir12]$ hw12
Enter the choice: (1 : search, 2 : add, 3 : delete, 4 : show, 0 : exit)
 1
Enter node to search: 18
19 -> 10 -> 14 -> 18
Enter the choice: (1 : search, 2 : add, 3 : delete, 4 : show, 0 : exit)
 2
Enter a new value: 2
19       left : 10       right : 46
10       left : 4        right : 14
46       left : 37       right : 55
4        left : 2        right : 7
14       left : 12       right : 18
37       left : 28       right : 40
55       left : 51       right : 61
2        left : empty    right : empty
7        left : empty    right : empty
12       left : empty    right : empty
18       left : empty    right : empty
28       left : 21       right : 32
40       left : empty    right : empty
51       left : 49       right : empty
61       left : 58       right : empty
21       left : empty    right : empty
32       left : empty    right : empty
49       left : empty    right : empty
58       left : empty    right : empty
```

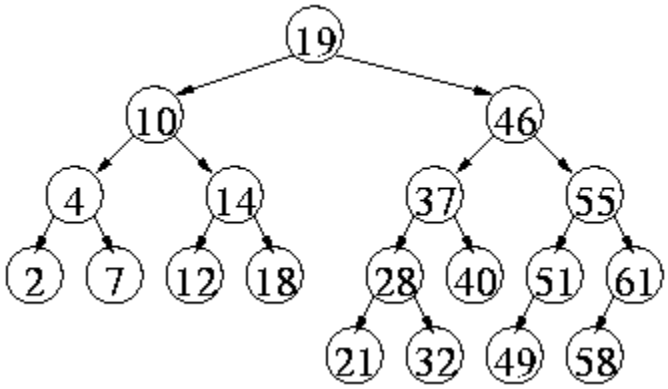
```
Enter the choice: (1 : search, 2 : add, 3 : delete, 4 : show, 0 : exit)
 3
Enter node to delete: 19
18        left : 10        right : 46
10        left : 4         right : 14
46        left : 37        right : 55
4         left : 2         right : 7
14        left : 12        right : empty
37        left : 28        right : 40
55        left : 51        right : 61
2         left : empty     right : empty
7         left : empty     right : empty
12        left : empty     right : empty
28        left : 21        right : 32
40        left : empty     right : empty
51        left : 49        right : empty
61        left : 58        right : empty
21        left : empty     right : empty
32        left : empty     right : empty
49        left : empty     right : empty
58        left : empty     right : empty
Enter the choice: (1 : search, 2 : add, 3 : delete, 4 : show, 0 : exit)
 1
Enter node to search: 18
18
Enter the choice: (1 : search, 2 : add, 3 : delete, 4 : show, 0 : exit)
 0

        Thank your for using AVL tree program
```

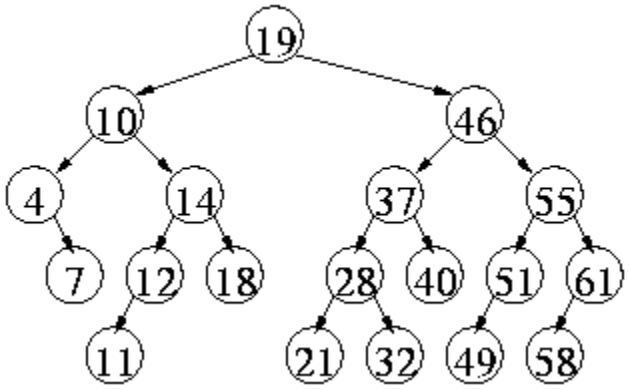
2. AVL Tree 연산 examples



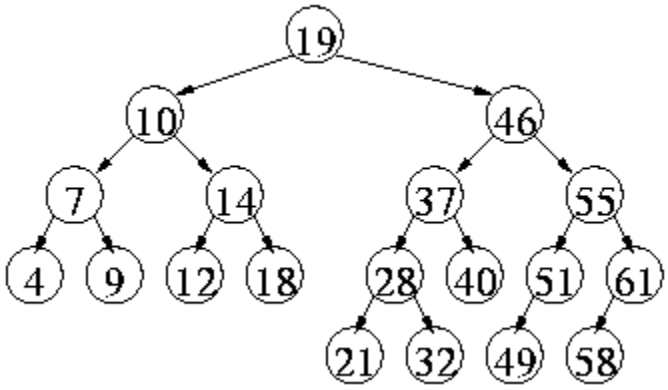
(Insert 2)



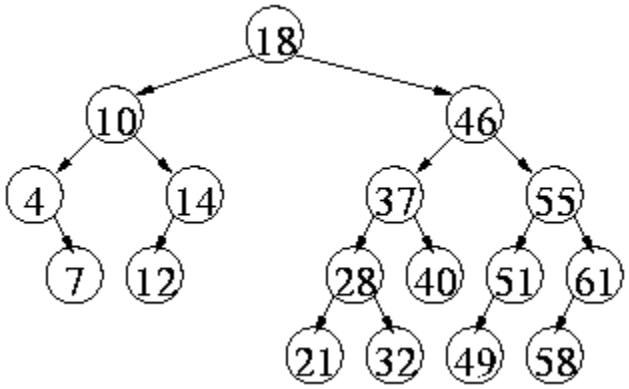
(Insert 11)



(Insert 9)



(Delete 19)



3. 코드

a. hw9.cpp

```
#include <iostream>

#include "BSTNode.h"
#include "BinarySearchTree.h"

using namespace std;

template <typename T>
void visit(const std::shared_ptr<BSTNode<T>> ptr) {
  //TODO:
}

int main() {
  BinarySearchTree<int> bst;
  bool flag = true;
  bst.insert(19); bst.insert(10); bst.insert(46);
  bst.insert(4);  bst.insert(14); bst.insert(37);
  bst.insert(55); bst.insert(7);  bst.insert(12);
  bst.insert(18); bst.insert(28); bst.insert(40);
  bst.insert(51); bst.insert(61); bst.insert(21);
  bst.insert(32); bst.insert(49); bst.insert(58);
  while (flag == true) {
    cout << "Enter the choice: (1 : search, 2 : add, 3 : remove, 4 : show, 0 :
            exit) ";
    int choice;
    cin >> choice;
    switch (choice) {
      case 1:
        cout << "Enter node to search: ";
        int toFind;
        cin >> toFind;
        bst.search(toFind);
        break;
      case 2:
        cout << "Enter a new value: ";
        int newValue;
        cin >> newValue;
        bst.insert(newValue);
        bst.show(visit<int>);
        break;
      case 3:
```

```
        cout << "Enter node to remove: ";
        int toRemove;
        cin >> toRemove;
        bst.remove(toRemove);
        bst.show(visit<int>);
        break;
      case 4:
        bst.show(visit<int>);
        break;
      case 0:
        cout << "\n\tThank your for using AVL tree program\n" << endl;
        flag = false;
        break;
      default:
        cout << "Sorry! wrong input\n" << endl;
        break;
    }
  }
  return 0;
}
```

b. AVL.h // 자유롭게 구현

```
#ifndef __BINARY_SEARCH_TREE_H__
#define __BINARY_SEARCH_TREE_H__
#include <memory>

#include "BSTNode.h"

template <typename T>
class BinarySearchTree {
  using TreePtr = std::shared_ptr<BSTNode<T>>;

 private:
  TreePtr rootPtr;

  /*
  Protected Utility Section:
  Recursive helper methods for the public methods.
  */
 protected:
  int getHeightHelper(TreePtr subTreePtr) const;
  int getNumberOfNodesHelper(TreePtr node) const;
  void clearHelper(TreePtr node);

  // 재귀 순회 헬퍼 함수
  void showHelper(std::function<void(TreePtr)> visit, TreePtr treePtr) const;

 public:
  /*
  생성자 및 소멸자
  */
  BinarySearchTree();
  BinarySearchTree(const T& rootItem);
  BinarySearchTree(const T& rootItem, const TreePtr leftTreePtr,
                   const TreePtr rightTreePtr);
  //   BinarySearchTree(const BinarySearchTree<T>& aTree);
  //   ~BinarySearchTree();

  /*
  Interface
  */
  bool isEmpty() const;
  int getHeight() const { return getHeightHelper(rootPtr); }
  int getNumberOfNodes() const;
  T getRootData() const { return rootPtr->getItem(); };
  void setRootData(const T& newData);
```

```
  // 교재에는 add
  bool insert(const T& newData);
  bool remove(const T& data);
  // 탐색 경로를 반환하는 함수
  // 출력 예시 > 10 -> 14 -> 19
  // 없을 경우 예시 > 10 -> 14 -> 19 -> Not Found
  void search(const T& data) const;
  void clear();
  T getEntry(const T& data) const;
  bool contains(const T& data) const;
  /*
  순회 함수
  */
  void show(std::function<void(TreePtr)> visit) const {
    showHelper(visit, rootPtr);
  };

  // AVL 관련 함수
  int getNodeHeight(TreePtr node) const;
  int getBalanceFactor(TreePtr node) const;
  TreePtr rotateLeft(TreePtr node);
  TreePtr rotateRight(TreePtr node);
  TreePtr balance(TreePtr node);
  TreePtr insertHelper(TreePtr node, const T& newData, bool& success);
  TreePtr removeHelper(TreePtr node, const T& data, bool& success);
  TreePtr findMin(TreePtr node) const;
};

#include "BinarySearchTree.cpp"

#endif  // __BINARY_SEARCH_TREE_H__
```

c. BSTNode.h

```
#ifndef __BSTNODE_H__
#define __BSTNODE_H__
#include <memory>

template <typename T>
class BSTNode {
 private:
  T item;
  std::shared_ptr<BSTNode<T>> leftChildPtr;
  std::shared_ptr<BSTNode<T>> rightChildPtr;
  int height;

 public:
  BSTNode();
  BSTNode(const T& anItem);
  BSTNode(const T& anItem, std::shared_ptr<BSTNode<T>> leftPtr,
          std::shared_ptr<BSTNode<T>> rightPtr);

  void setItem(const T& nodeItem);
  T getItem() const;

  void setLeftChildPtr(std::shared_ptr<BSTNode<T>> leftPtr);
  void setRightChildPtr(std::shared_ptr<BSTNode<T>> rightPtr);

  auto getLeftChildPtr() const;
  auto getRightChildPtr() const;

  void setHeight(int h) { height = h; }
  int getHeight() const { return height; }
};  // end BSTNode

#include "BSTNode.cpp"
#endif  // __BSTNODE_H__
```

d. BinarySearchTree.cpp, BSTNode.cpp 구현