HW12 과제: AVL 트리

제출 데드라인: 과제 부여 후 2주 후 24시 까지
목요일 분반: 12월 15일 목요일 24시 까지 (12월 16일 금요일 0시 까지)
금요일 분반: 12월 16일 금요일 24시 까지 (12월 17일 토요일 0시 까지)

제출 명령어: submit pem_ta hw12x (x=a,b,c)
제출 확인: submit pem_ta hw12x -l (마이너스 엘)

공지사항
a. 소스파일(헤더파일 포함), 결과보고서를 함께 리눅스에 submit
b. 결과보고서는 과제의 이해도를 평가하는 중요한 척도입니다. 구현 과정과 분석결과를 최대한 자세히 작성해주세요.

질문
pemta81718@gmail.com

제출파일
AVL.h AVL.cpp hw12.cpp 학번.tex 학번.pdf
+ 보고서에 첨부한 이미지 파일들
// 정확한 파일명은 중요합니다

보고서 내용
1. Class 설계 내용 및 이유
2. 결과 값 (출력) 및 결과 분석
3. 결과 값 (출력) 스크린 샷
4. AVL Tree 의 연산들 설명
   - 특히, 4 종류의 회전 연산에 대한 설명 (그림 및 코드 주석 첨부)
5. 어려웠던 부분 등...

주의사항

a. 조교의 전달사항을 잘 따를 것. (미 출석하신 분은 따로 메일)

b. Cheating 은 F.

c. 리눅스가 12 시간 이상 오류 나지 않는다면 기간 연장은 없다.

d. 기본적으로 메일로 제출된 과제는 읽지 않는다.

e. 제출 마감시간 기준으로 4 시간 전부터 마감까지 제출이 불가능하다면, 이메일로 제출도 받는다.

f. 하지만, 제출 마감시간 15분 전부터 마감시간까지 제출이 가능했다면, 이메일로 제출된 것은 읽지 않는다.

1. 실습 및 과제 내용

hw12.cpp 와 아래의 실행 예시를 참고하여 간단한 AVL Tree 를 구현한다.

AVL.cpp 의 빈칸을 채워 주어진 함수들을 연산할 수 있는 AVL Tree 를 설계한다.

Node struct 또한 자유롭게 정의한다.

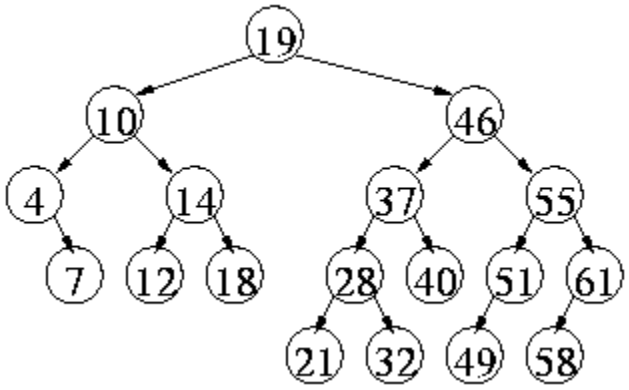
실행 예시:

```
[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
```

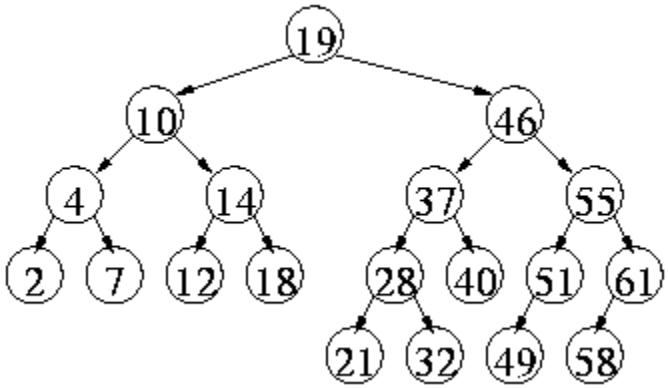
```
Enter the choice: (1 : search, 2 : add, 3 : delete, 4 : show, 0 : exit)
 3
Enter node to delete: 19
18        left : 10        right : 46
10        left : 4         right : 14
46        left : 37        right : 55
4         left : 2         right : 7
14        left : 12        right : empty
37  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

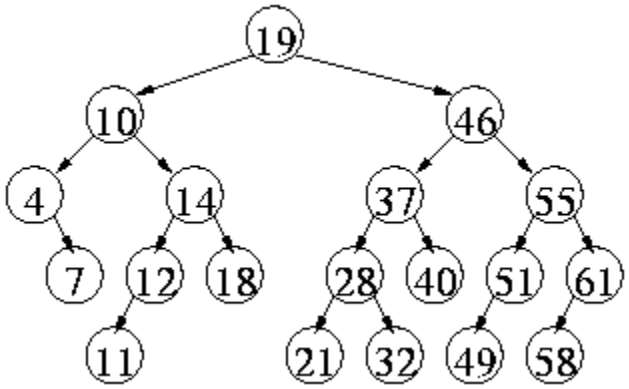
2. AVL Tree 연산 examples



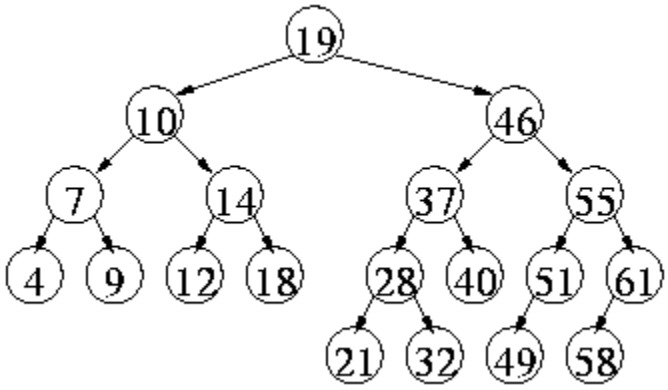
(Insert 2)



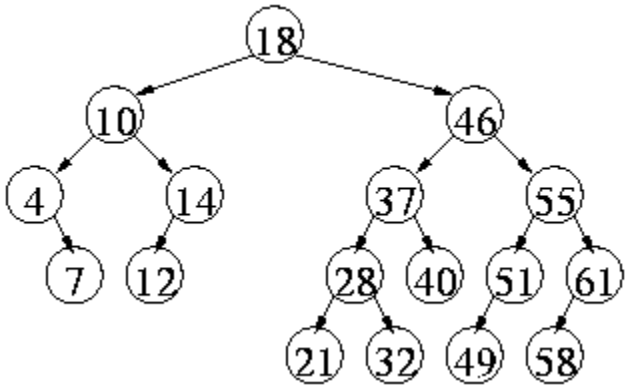
(Insert 11)



(Insert 9)



(Delete 19)



3. 코드

a. makefile

명령어: cat makefile / vi makefile

hw12: (tab) hw12.o avl.o

(tab) g++ -o hw12 hw12.o avl.o

b. hw12.cpp

```
#include "AVL.h"
int main()
{
	nodeptr root;
	int a, choice, findele, delele;
	bstree bst;
	bool flag = true;
	root = NULL;
	bst.insert(19, root); bst.insert(10, root); bst.insert(46, root);
	bst.insert(4, root); bst.insert(14, root);
	bst.insert(37, root); bst.insert(55, root); bst.insert(7, root);
```

```
	bst.insert(12, root); bst.insert(18, root);
	bst.insert(28, root); bst.insert(40, root); bst.insert(51, root);
	bst.insert(61, root); bst.insert(21, root);
	bst.insert(32, root); bst.insert(49, root); bst.insert(58, root);
	while (flag == true)
	{
		cout << "Enter the choice: (1 : search, 2 : add, 3 : delete, 4 : show, 0 :
exit) ";
			cin >> choice;
		switch (choice)
		{
		case 1:
			cout << "Enter node to search: ";
			cin >> findele;
			if (root != NULL)
				bst.Search(findele, root);
			break;
		case 2:
			cout << "Enter a new value: ";
			cin >> a;
			bst.insert(a, root);
			bst.Showresult(root);
			break;
		case 3:
			cout << "Enter node to delete: ";
			cin >> delele;
			bst.del(delele, root);
			bst.Showresult(root);
			break;
		case 4:
			if (root != NULL)
```

```
                        bst.Showresult(root);
                    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;
}
```

c. AVL.h

```
#ifndef AVL_H
#define AVL_H
typedef struct Node* nodeptr;
#include <queue>
#include <iostream>
using namespace std;

struct Node {

};

class bstree {

};
#endif
```

d. AVL.cpp

```
#include "AVL.h"

void bstree::visit(Node* ptr) {
	cout << ptr->data << "₩t";
	if (ptr->leftChild) { cout << "left : " << ptr->leftChild->data << '₩t'; }
	else { cout << "left : empty₩t"; }
	if (ptr->rightChild) { cout << "right : " << ptr->rightChild->data << '₩t'; }
	else { cout << "right : empty₩t"; }
	cout << endl;
}
void bstree::insert(const int value, Node*& root) {

};
void bstree::show(Node* root) {

};
bool bstree::search(const int key, Node* root) {

}
void bstree::del(const int key, Node* root) {

};
void bstree::setBF(Node* startNode, Node* endNode) {

}
bool bstree::rotation(Node* start, Node* end) {

}
```