Data Structure Assignment 6

ID: E14066282	**Name:** 溫梓傑	**Department:** ME 110

○ Result Screenshots

```
Wun@DESKTOP-1MSRHTB MINGW64 /c/2020-NCKU_DS/HW6_AVL_tree/code (master)
$ gcc -std=c11 ./*.c -o hw6

Wun@DESKTOP-1MSRHTB MINGW64 /c/2020-NCKU_DS/HW6_AVL_tree/code (master)
$ ./hw6.exe < input0_windows.txt > ans_output0_windows.txt

Wun@DESKTOP-1MSRHTB MINGW64 /c/2020-NCKU_DS/HW6_AVL_tree/code (master)
$ diff ./output0_windows.txt ./ans_output0_windows.txt
```

Figure 1 Screenshot of command line

ans_output0_windows.txt

code > ans_output0_windows.txt

```
1  David Bob Alice Charlie Paul Ruby
2  Paul 0900000002
3  Amy null
4  Ruby 0900000004
```

Figure 2 ans_output0_windows.txt

○ Program Architecture

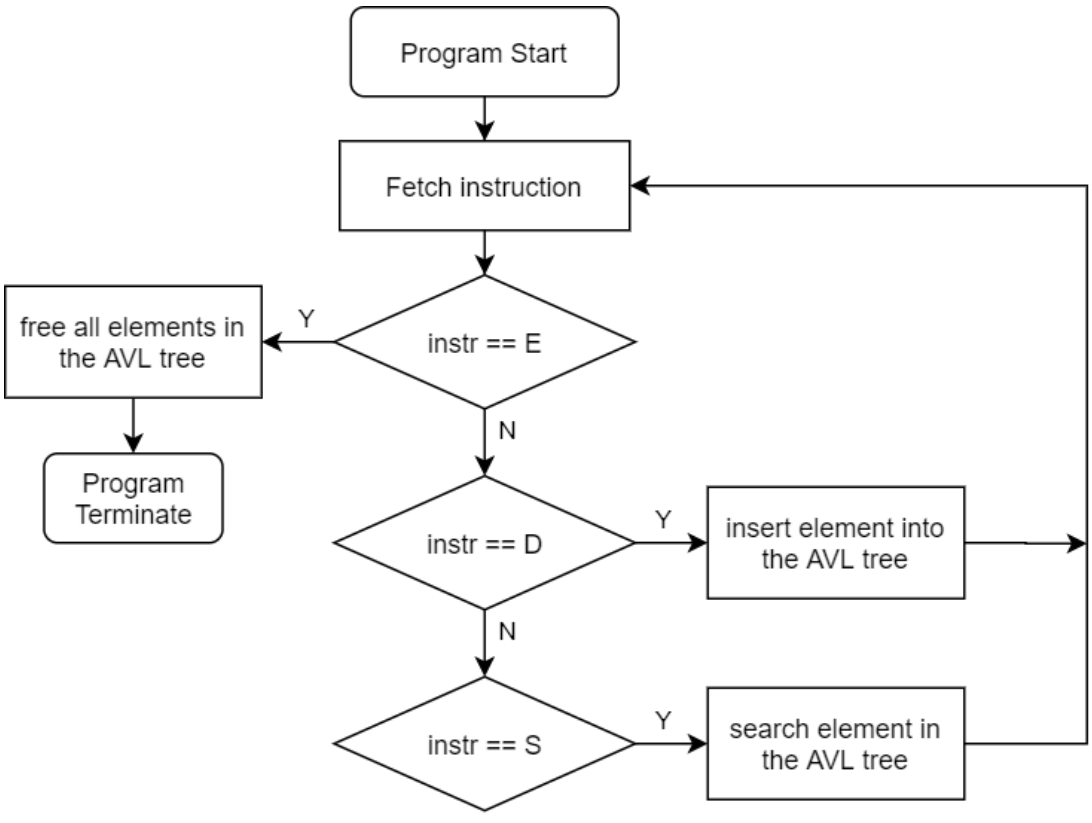

Figure 3 Flow chart of hw6

○ Program Functions

- 📒 **AVL_Tree.h**

```
Node_tr *create_node_tr(char name[21], char phone[11])
```

Constructs a tree node.

- 📐 **Parameters**

```
name, phone
```

The element that would be initialized in the constructed tree node.

- ⮌ **Return Value**

Returns the new pointer of the node.

  - If construction fails, returns NULL.

```
Node_tr *insert_tree(Node_tr *root, char name[21], char phone[11])
```

- 📐 **Parameters**

```
root
```

The root node of the BST (binary search tree).

```
name, phone
```

The elements to be inserted to the BST.

- ⮌ **Return Value**

The new tree pointer that it is modified.

```
void update(Node_tr *node)
```

Update balance factor and height of the node.

- 📐 **Parameters**

```
node
```

The node to be updated.

- ⮌ **Return Value**

None.

```
void balance(Node_tr *root)
```

Balance the tree if it is unbalanced (balance factor equals to 2 or -2).

- **Parameters**

```
root
```

The root to be balanced.

- **Return Value**

None.

```
Node_tr *L_rot(Node_tr *node)
```

Do a left rotation to the node of the tree.

- **Parameters**

```
node
```

The root node to be rotated.

- **Return Value**

The new root pointer if it is modified.

```
Node_tr *R_rot(Node_tr *node)
```

Do a right rotation to the node of the tree.

- **Parameters**

```
node
```

The root node to be rotated.

- **Return Value**

The new root pointer if it is modified.

```
Node_tr *LR_rot(Node_tr *node)
```

Do a left rotation to its left child and then do a right rotation to its root node.

- **Parameters**

```
node
```

The root node to be rotated.

- **Return Value**

The new root pointer if it is modified.

```
Node_tr *LR_rot(Node_tr *node)
```

Do a right rotation to its right child and then do a left rotation to its root node.

- **Parameters**

`node`

The root node to be rotated.

- **Return Value**

The new root pointer if it is modified.

```
void print_tree_PRE(FILE *fp, Node_tr *root, int first)
```

Print the elements in the tree with pre-order traversal (VLR).

- **Parameters**

`fp`

The file stream to be printed.

`node`

The root node to be printed.

`first`

The integer that can decide whether it is root node. Fill it with 0 always.

- **Return Value**

None.

```
void free_tree(Node_tr *root)
```

Free all elements in the AVL tree.

- **Parameters**

`node`

The root node of the tree to be freed.

- **Return Value**

None.

○ Program Design

• Structure of AVL tree node

為了實現 AVL tree，因此新增兩個參數：height、balance factor。

```
typedef struct Node_tr {
    struct Node_tr *left;
    struct Node_tr *right;
    char name[21];  // key
    char phone[11]; // value
    // AVL tree parameters
    int height;
    int bf; // balance factor
} Node_tr;
```

• Insertion of AVL tree

```
Node_tr *insert_tree(Node_tr *root, char name[21], char phone[11]) {
    if (!root)
        return create_node_tr(name, phone);
    if (strcmp(name, root->name) < 0)
        root->left = insert_tree(root->left, name, phone);
    else
        root->right = insert_tree(root->right, name, phone);
    update(root);
    return balance(root);
}
```

使用遞迴方式進行插入，如此一來可以追蹤至 leaf node 後，再反向追蹤，一一將不平衡的部分進行旋轉的動作。

- **Rotation**

1. LL case: right rotation

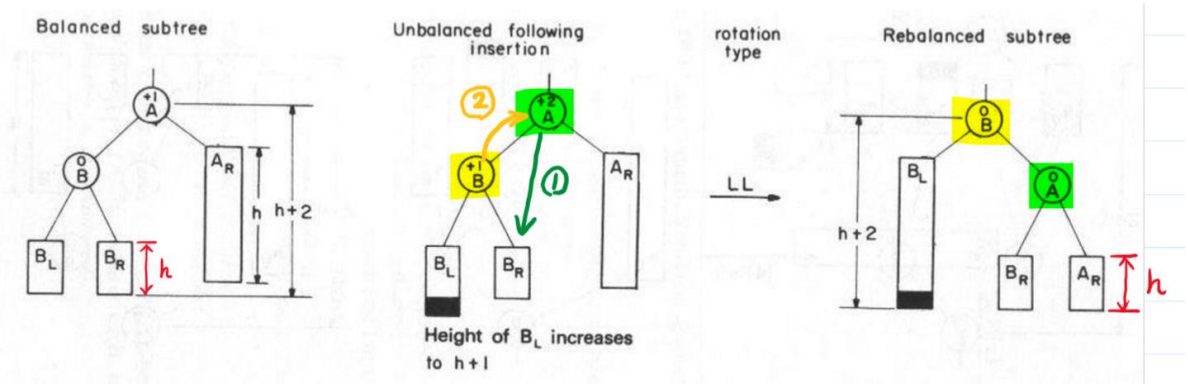

Figure 4 Details of right rotation

2. RR case: left rotation

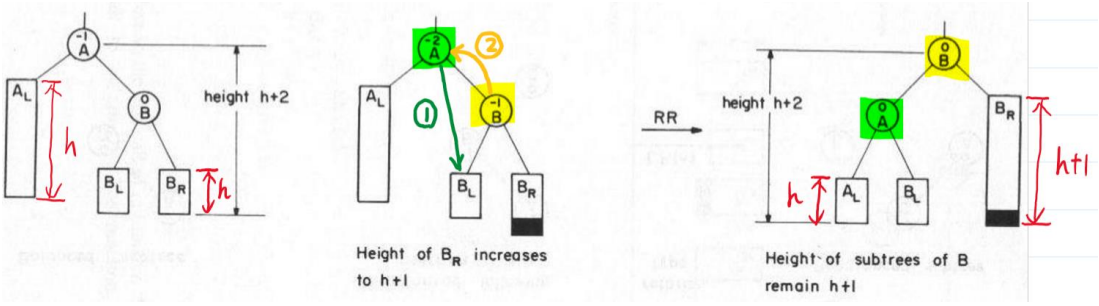

Figure 5 Details of left rotation

3. LR case: left rotation at B → right rotation at A

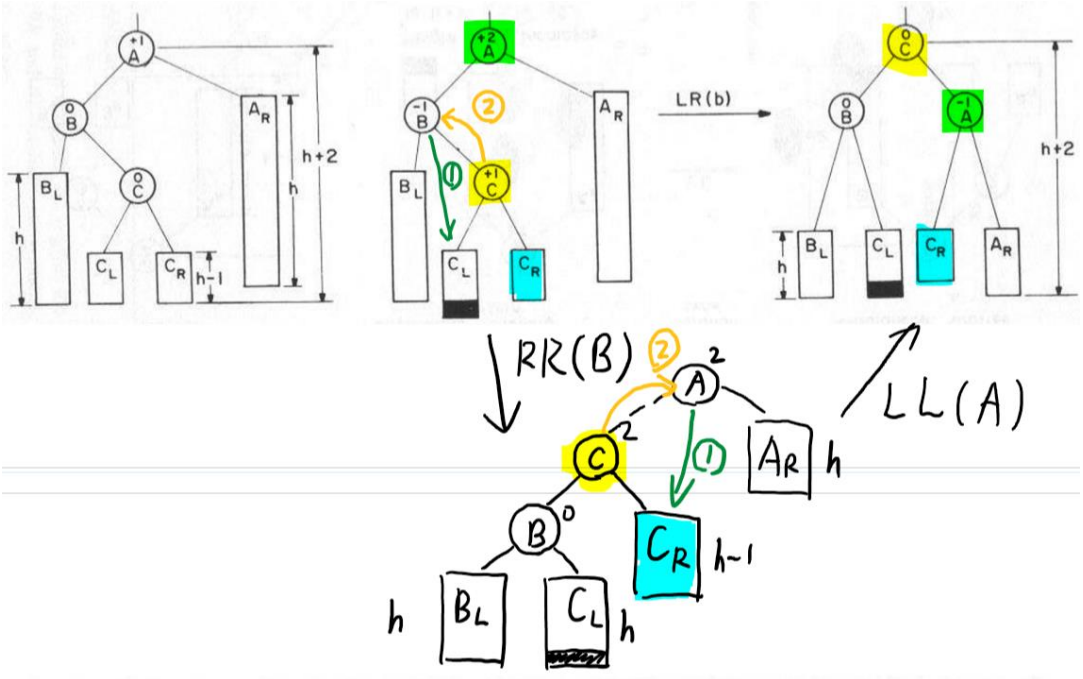

Figure 6 Details of LR case

4. RL case: 與 LR case 同理，左右相反即可。

- **Print AVL tree in Pre-Order**

```
void print_tree_PRE(FILE *fp, Node_tr *root, int first) {
    if (!root)
        return;
    // Visit
    fprintf(fp, "%s%s", first == 0 ? "" : " ", root->name);
    if (root->left)
        print_tree_PRE(fp, root->left, 1);
    if (root->right)
        print_tree_PRE(fp, root->right, 1);
}
```

以上為經典教科書中常見的 pre-order 造訪方法，唯一較為特殊的地方在於，多了一個 `first` 參數，其目的是為了不要讓第一個被 print 出來的 node，多出一個 space character，僅此而已。

- **Operating System**
  Windows 10
- **Compiler**
  gcc.exe (MinGW.org GCC Build-20200227-1) 9.2.0
- **Compile**

```
gcc -std=c11 ./*.c -o hw6
```

- **Run**

```
./hw6.exe < input.txt > output.txt
```