# Data Structure Assignment 6

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

## Result Screenshots

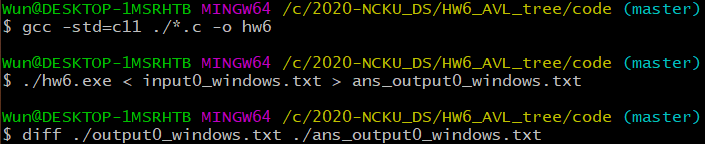


Figure Screenshot of command line

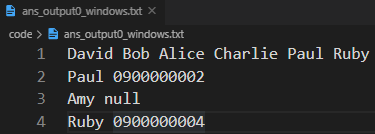


Figure ans\_output0\_windows.txt

## Program Architecture

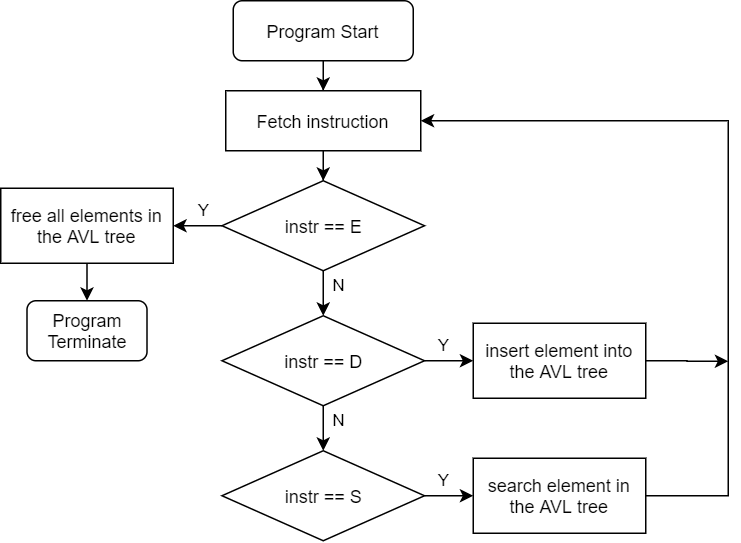


Figure Flow chart of hw6

## 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

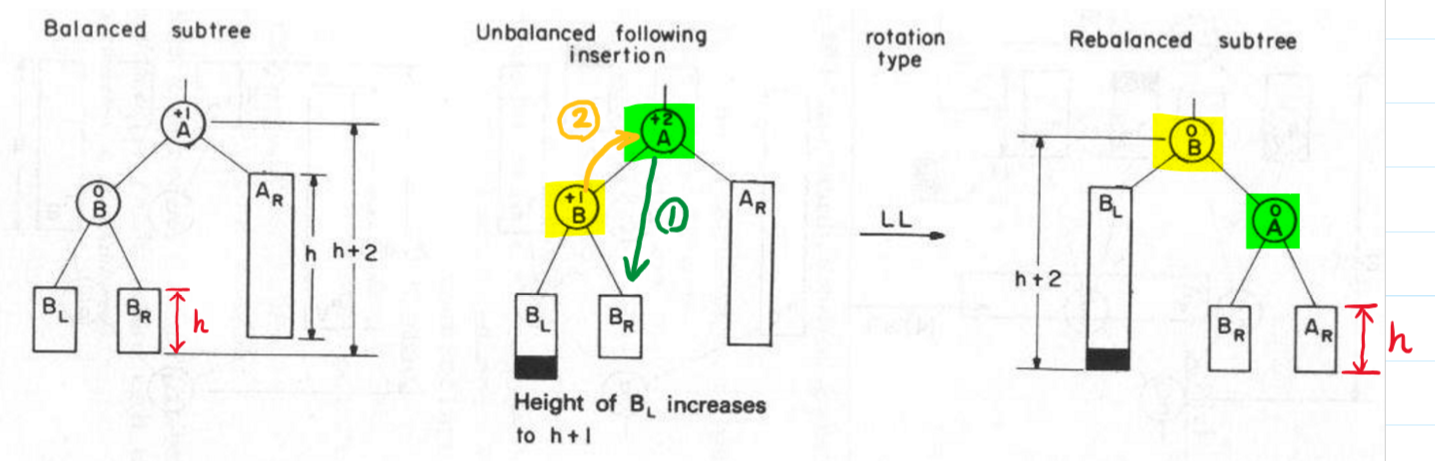


Figure Details of right rotation

1. RR case: left rotation

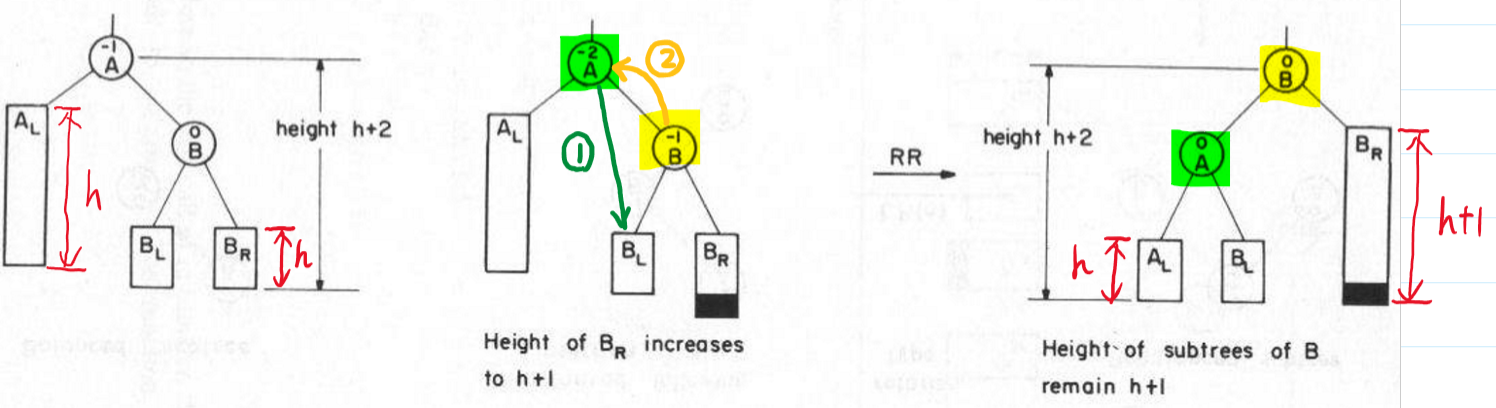


Figure Details of left rotation

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

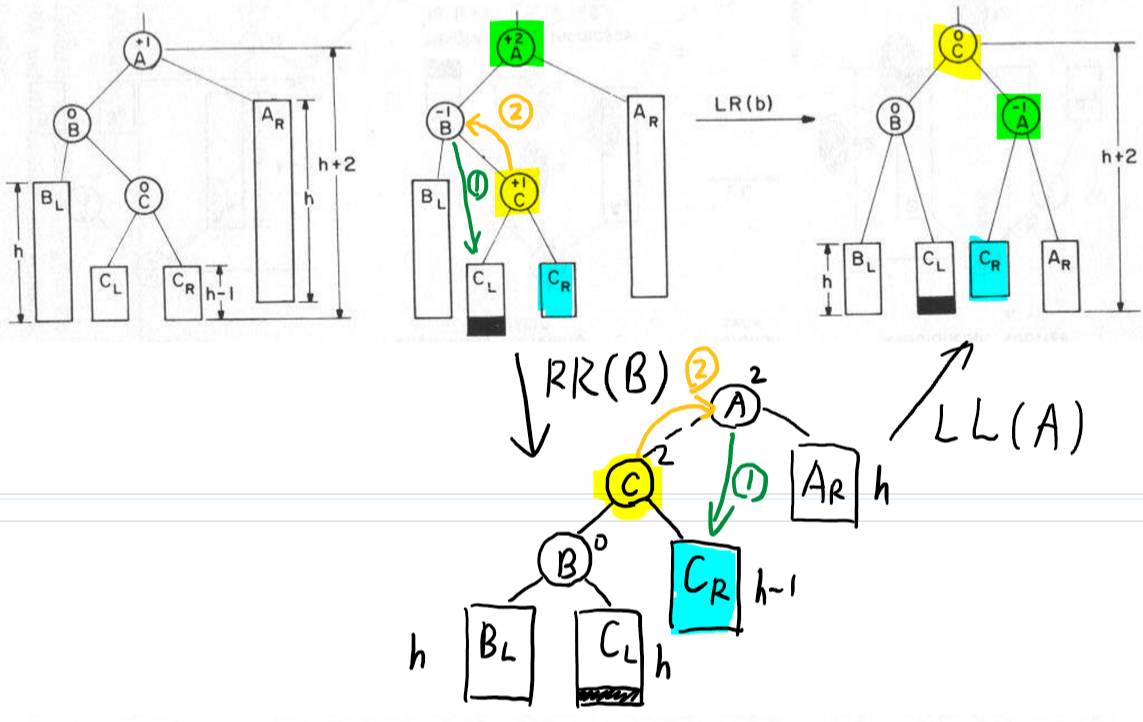


Figure Details of LR case

1. 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