

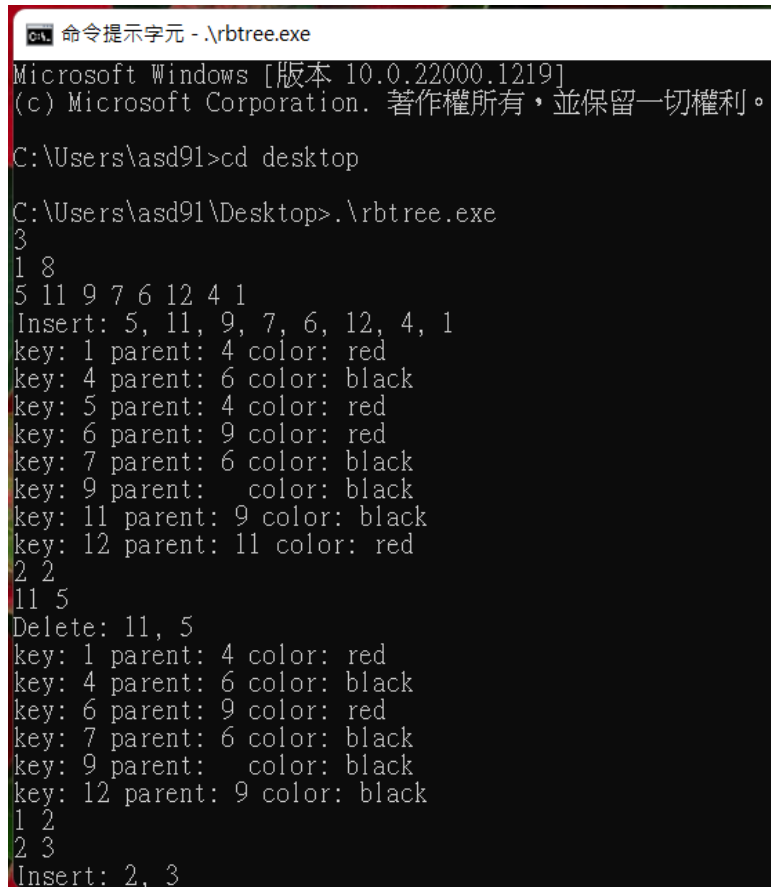
- Environment:

OS: Windows

Compiler: gcc/g++ version: 11.2.0

IDE: Visual Studio (version: 16.11.5)

可以用 cmd 開啟後執行, 直接輸入 input



```
命令提示字元 - .rbtree.exe
Microsoft Windows [版本 10.0.22000.1219]
(c) Microsoft Corporation. 著作權所有，並保留一切權利。
C:\Users\asd91>cd desktop
C:\Users\asd91\Desktop>.\rbtree.exe
3
1 8
5 11 9 7 6 12 4 1
Insert: 5, 11, 9, 7, 6, 12, 4, 1
key: 1 parent: 4 color: red
key: 4 parent: 6 color: black
key: 5 parent: 4 color: red
key: 6 parent: 9 color: red
key: 7 parent: 6 color: black
key: 9 parent:    color: black
key: 11 parent: 9 color: black
key: 12 parent: 11 color: red
2 2
11 5
Delete: 11, 5
key: 1 parent: 4 color: red
key: 4 parent: 6 color: black
key: 6 parent: 9 color: red
key: 7 parent: 6 color: black
key: 9 parent:    color: black
key: 12 parent: 9 color: black
1 2
2 3
Insert: 2, 3
```

也可以用 visual studio 開啟編譯後直接輸入 input

```

C:\Users\asd91\Desktop\109350008_HW2\Debug\109350008_HW2.exe
3
1 8
5 11 9 7 6 12 4 1
Insert: 5, 11, 9, 7, 6, 12, 4, 1
key: 1 parent: 4 color: red
key: 4 parent: 6 color: black
key: 5 parent: 4 color: red
key: 6 parent: 9 color: red
key: 7 parent: 6 color: black
key: 9 parent:    color: black
key: 11 parent: 9 color: black
key: 12 parent: 11 color: red
2 2
11 5
Delete: 11, 5
key: 1 parent: 4 color: red
key: 4 parent: 6 color: black
key: 6 parent: 9 color: red
key: 7 parent: 6 color: black
key: 9 parent:    color: black
key: 12 parent: 9 color: black
1 2
2 3
Insert: 2, 3
key: 1 parent: 2 color: black
key: 2 parent: 6 color: red
key: 3 parent: 4 color: red
key: 4 parent: 2 color: black
key: 6 parent:    color: black
key: 7 parent: 9 color: black

```

## ● Result:

這次思路以及邏輯主要是參考老師的 ppt 以及課本的 pseudocode 但是再做一些些微調，像是 left、right rotate 裡面直接使用 transparent，還有 deletion 先做找節點的動作。雖然最後助教有註明說不會刪除不存在的節點還有不會有相同 key 值存在，但我還是做了防呆(碰到上述狀況直接 return)。

紅黑樹 insert 和 delete:

### Insert:

插入 key 的規則和 binary search tree 一樣，但是每個節點新增後一定是紅色，之後再視情況調整至符合紅黑樹特性：

- 若是插入為根節點，直接染黑。
- 若插入後其 parent 節點為黑色，甚麼都不做，但若為紅色，則再看新增節點的 uncle 決定：

若新增節點的 parent 為左節點：

1. 若其 uncle 為紅色

對策: parent、uncle 塗黑, grandparent 塗紅, 當前節點指向 grandparent

2. 若其 uncle 是黑色, 新增節點為右節點

對策: 對 parent 左旋

3. 若其 uncle 是黑色, 新增節點為左節點

對策: Parent 塗黑, grandparent 塗紅, 對 grandparent 右旋  
左右交換再打一次

最後再把根節點塗黑。

### Delete:

先找到要刪除的節點, 接著按照要刪除節點的 child 情況做調整:

- 無 child: 其 parent 的 child 指向 T.nil
- 只有一個 child: 其 parent 的 child 指向其 child
- 兩個 child: 找 successor 或是 predecessor 取代刪除節點

若是刪除節點為黑色則有可能會因此違反紅黑樹特性, 因此需要再視刪除後的情況做調整:

若當前節點為左節點:

1. Sibling 為紅色:

對策: sibling 塗黑, parent 塗紅並且做 left rotate

2. Sibling 為黑色, 且其 2 個 child 都為黑色

對策: sibling 塗紅, 當前節點移向 parent

3. Sibling 為黑色, 且其 right child 為黑色, left child 為紅色

對策: sibling 的 left child 塗黑, sibling 塗紅並且做 right rotate

4. Sibling 為黑色, right child 為紅色

對策: sibling 塗成當前節點 parent 的顏色, parent 塗黑, sibling 的 right child 塗黑, 對 parent 做 left rotate, 當前節點移向根節點

左右交換再做一次

最後把根節點塗黑

以下是我的微調:

\*left\_rotate:

```

void RBT::transparent(node* transed, node* transing)
{
    if (transed->parent == nil) root = transing;
    else if (transed == transed->parent->left_child) transed->parent->left_child = transing;
    else transed->parent->right_child = transing;
    transing->parent = transed->parent;
}

void RBT::left_rotate(node* cur)
{
    node* y = cur->right_child;

    cur->right_child = y->left_child;
    if (y->left_child != nil) y->left_child->parent = cur;
    transparent(cur, y);
    y->left_child = cur;
    cur->parent = y;
}

```

\*delete 找節點以及刪除空值的防呆:

```

void RBT::deletion(int key)
{
    //find
    node* d_node = new node;
    d_node->key = key;
    node* x = root;

    while (x != nil)
    {
        if (x->key < d_node->key) x = x->right_child;
        else if (x->key > d_node->key) x = x->left_child;

        if (x->key == d_node->key)
        {
            d_node = x;
            break;
        }
    }
    if (x == nil) return;
}

```

\*insert 以存在或是相同的 key 的防呆:

```

void RBT::insertion(int key)
{
    node* y = nil;
    node* x = root;
    node* new_node = new node;

    new_node->key = key;
    while (x != nil)
    {
        y = x;
        if (x->key < new_node->key) x = x->right_child;
        else if (x->key > new_node->key) x = x->left_child;
        else return;
    }
}

```

這次的報告有點難打，因為主要的想法都在老師的 ppt 或是課本裡面，我已經把能打得盡量打了 QQ，謝謝助教閱讀。