```
#ifndef BSTlib_h
#define BSTlib_h

struct node
{
    struct node *left;
    struct node *right;
};
typedef struct node node_t;
```

以上為使用者可使用 node_t 的結構。

若使用此 library，可使用到以下的功能

（１）插入一個節點進 BST（其插入的每個值必須符合 BST 的定義，即 BST 中的值不可有重複）

void insertNode(void *element, node_t **root, int (*cp)(void *A, void *B));

傳入值：

第一個參數：型態為 void，想要放入樹(root)節點的指標

第二個參數：型態為 node_t，樹根（root)指標的位址

第三個參數：函式指標。

需使用者寫一個可以比較兩個 node 的函式並傳入，如：

```
int comp(void *node1, void *node2)
{
    if( ((student*)node1)->grade > ((student*)node2)->grade )
        return 1;
    else if( ((student*)node1)->grade < ((student*)node2)->grade )
        return -1;
    else
        return 0;
}
```

（以上為欲比較學生成績的函式）

而需要其傳回的結果符合以下幾點

(1)當 node1 的值 > node2 的值時，回傳 1

(2)當 node1 的值 < node2 的值時，回傳-1

(3)當 node1 的值 = node2 的值時，回傳 0

回傳值：無回傳值

（２）刪除 BST 中，與第一個傳入參數相等的值

node_t *deleteNode(void *target, node_t **root, int (*cp)(void *A, void *B));

傳入值：

第一個參數：型態為 void，在 tree(樹)中刪除與傳入參數相同的值的指標

第二個參數：型態為 node_t，樹根(root)指標的位址

第三個參數：函式指標。

需使用者寫一個可以比較兩個值(target&root 中的值)的函式並傳入，如：

```
int comp(void *node1, void *node2)
{
    if( ((student*)node1)->grade > ((student*)node2)->grade )
        return 1;
    else if( ((student*)node1)->grade < ((student*)node2)->grade )
        return -1;
    else
        return 0;
}
```

（以上為欲比較學生成績的函式）

而需要其傳回的結果符合以下幾點

(1)當 node1 的值 > node2 的值時，回傳 1

(2)當 node1 的值 < node2 的值時，回傳-1

(3)當 node1 的值 = node2 的值時，回傳 0

回傳值：無回傳值

（ 3 ）找出 BST 中最小的鍵值

node_t *findMinNode(node_t *root);

傳入值：型態為 node_t，樹根(root)的指標

回傳值：型態為 node_t，指向最小值節點的指標

（ 4 ）找出 BST 中最大的鍵值

node_t *findMaxNode(node_t *root);

傳入值：型態為 node_t，樹根(root)的指標

回傳值：型態為 node_t，指向最小值節點的指標

（ 5 ）找出 BST 符合鍵值的節點

node_t *findNode(void *target, node_t *root, int (*cp)(void *A, void *B));

傳入值：

第一個參數：型態為 void，想找到在此樹中與此參數同值的節點的指標

第二個參數：型態為 node_t，樹根(root)指標的位址

第三個參數：函式指標。

需使用者寫一個可以比較兩個值(target&root 中的值)的函式並傳入，如：

```
int comp(void *node1, void *node2)
{
    if( ((student*)node1)->grade > ((student*)node2)->grade )
        return 1;
    else if( ((student*)node1)->grade < ((student*)node2)->grade )
        return -1;
    else
        return 0;
}
```

（以上為欲比較學生成績的函式）

而需要其傳回的結果符合以下幾點

(1)當 node1 的值 > node2 的值時，回傳 1

(2)當 node1 的值 < node2 的值時，回傳-1

(3)當 node1 的值 = node2 的值時，回傳 0

回傳值：型態為 node_t，指向和 target 相同值，在 root 中的指標

（ 6 ）根據中序追蹤法，列印出 BST 中每個節點的內容

void inOrder(node_t *root, void(*print_node)(void*));

傳入值：

第一個參數：型態為 node_t，傳入想要列印的樹的樹根(root)的指標

第二個參數：傳入一個指標函數，此函式可以將 root 中你想要的值列印出來

回傳值：無回傳值

（ 7 ）複製 BST tree

void treeCopy(node_t **new_tree, node_t *root, int struct_node_size);

傳入值：

第一個參數：傳入一個 node_t 型態，要複製到的樹的樹根的指標的位置

第二個參數：傳入一個 node_t 型態，要被複製的樹的樹根的指標

第三個參數：傳入一個 sizeof(root 中節點的 struct)後的 int 值

回傳值：無回傳值

（ 8 ）比較兩個 BST 是否相同

void treeEqual(node_t *tree1, node_t *tree2, int(*cp)(void *A, void *B), int *flag);

傳入值：

第一個參數：傳入一個 node_t 型態，要被比較的樹的樹根

第二個參數：傳入一個 node_t 型態，要被比較的樹的樹根

第三個參數：函式指標。

需使用者寫一個可以比較兩個值(tree1 &tree2 中的值)的函式並傳入，如：

```
int comp(void *node1, void *node2)
{
    if( ((student*)node1)->grade > ((student*)node2)->grade )
        return 1;
    else if( ((student*)node1)->grade < ((student*)node2)->grade )
        return -1;
    else
        return 0;
}
```

（以上為欲比較學生成績的函式）

而需要其傳回的結果符合以下幾點

(1)當 node1 的值 > node2 的值時，回傳 1

(2)當 node1 的值 < node2 的值時，回傳-1

(3)當 node1 的值 = node2 的值時，回傳 0

第四個參數：需傳入一個 int 型態，且其值為 1 的位址。

回傳值：無回傳值

（使者最後可以在自己的 main 中判斷 flag 的值，flag = 1 表示兩個 tree 相等，flag = 0 表示兩個 tree 不相等）

#endif