函數使用說明書 (pdf 檔)

* 事前準備：
  1. #include "BST.h"
  2. 使用以下語法製作出執行檔

gcc main.c**(****你的程式碼)** libBST.a **(BST library檔)** -o main\_static**(輸出執行檔)**

* 1. 建立自己的結構(必須將btreeNode\_t結構放置到最上方)，如下顯示。

typedef struct myBST {

btreeNode\_t treeNode;

char ID[10];

int math;

int eng;

} student\_t;

* 1. 根據你的結構，定義清楚複製、比較與打印的函式，如下顯示:
     + void copy(void \*elementA, void \*elementB){

strncpy(((student\_t\*)elementA)->ID,((student\_t\*)elementB)->ID,10);

((student\_t \*)elementA)->math = ((student\_t \*)elementB)->math;

((student\_t \*)elementA)->eng = ((student\_t \*)elementB)->eng;}

* int compareID(void \*elementA, void \*elementB) {

char \*aid = ((student\_t \*)elementA)->ID;

char \*bid = ((student\_t \*)elementB)->ID;

for (int i=0;i<10;i++) {

if(aid[i]>bid[i]) {

return 1;

}else if(aid[i]<bid[i]){

return -1;}}

return 0;}

* void print\_node(void \*elemant){

if(elemant){

char \*id = ((student\_t \* )elemant)->ID;

int eng = ((student\_t \* )elemant)->eng;

printf("id = %s ",id);

printf("eng = %d",eng);

printf("\n");}

else

printf("this node is null!");}

* 1. 建立一棵樹根(treeRoot)

btreeNode\_t \* treeRoot = (btreeNode\_t\*)student\_t \*root;

* Binary Search Tree (BST) library 功能使用說明：

--------------------------------------------------------------------------------------------------------------

btreeNode\_t \* **BST\_insertNode**(void \* element, btreeNode\_t \* root, int(\*compare)(void \* elementA, void \* element));

* 函式說明: 插入一個節點進 BST 內，並且回傳 該BST的root位置。
* 輸入要求:
  1. element: 為要新增到指定樹中的一個節點，可由下方語法建立。

student\_t \*insert\_node = (student\_t\*)malloc(sizeof(student\_t));

strncpy(insert\_node->ID, "123",10); //給予ID值

insert\_node->eng = 100; //給予英文分數

* 1. btreeNode\_t \* root: 告訴lib要將element新增到**何處**。
  2. compare: 利用函數指標告訴lib要**如何**將element新增到樹中，也就是使用事前準備好的compareID函式，讓lib知道要以哪一個數值(此範例使用ID值)進行比較與新增節點。
* 使用範例:

treeRoot = BST\_insertNode((btreeNode\_t\*)insert\_node, treeRoot, compareID);

--------------------------------------------------------------------------------------------------------------

void **BST\_inOrder**(btreeNode\_t\*root,void(\*print\_node)(void\*element));

* 函式說明: inOrder 列印 BST 根據中序追蹤法每個節點內容。
* 輸入要求:

1. btreeNode\_t \* root: 輸入要inOrder 列印的 BST 樹根位置。
2. print\_node: 利用函數指標告訴lib要**如何**將追蹤到的節點印出來，也就是使用事前準備好的print\_node函式，讓lib知道需要印出該節點中的哪些內容。

* 使用範例:

printf("\ninOrder search:\n");

BST\_inOrder(treeRoot,print\_node);

* 輸出成果:

一張含有 文字, 電子用品 的圖片

自動產生的描述

--------------------------------------------------------------------------------------------------------------

------------------------------------------------------------------------btreeNode\_t \* BST\_findMinNode(btreeNode\_t \* root);

//findMinNode 找出 BST 中鍵值最小的節點

btreeNode\_t \* BST\_findMaxNode(btreeNode\_t \* root);

//findMaxNode 找出 BST 中鍵值最大的節點

btreeNode\_t \* BST\_findNode(void \* element, btreeNode\_t \* root, int

(\*compare)(void \* elementA, void \* element));

* 函式說明: 找出 BST root中與element鍵值相同的節點。
* 輸入要求:

1. element: 輸入所要尋找的節點。
2. btreeNode\_t \* root: 輸入要尋找的 BST 樹根位置。
3. compare: 利用函數指標告訴lib要**如何**在root 中尋找element節點。使用事前準備好的compareID函式，讓lib知道要以哪一個數值(此範例使用ID值)進行比較與收尋節點。

* 使用範例:

student\_t \*need = (student\_t\*)BST\_findNode(element, treeRoot, compareID);

printf("\nfind\_node(element):\n");

print\_node(need);

* 回傳結果:

如果有成功收尋到該節點，會回傳該節點在樹中的位置，提供使用者後續使用(修改、打印等等)；如果沒有收尋到該節點時，會回傳NULL並顯示**"no node!"**。

------------------------------------------------------------------------

btreeNode\_t \* **BST\_treeCopy**(btreeNode\_t \* root,void(\*copy)(void \* elementA, void \* element),int struct\_size);

* 函式說明: 將root位置的BST複製，並回傳新的樹根位置。
* 輸入要求:

1. btreeNode\_t \* root: 輸入要複製的 BST 樹根位置。
2. copy: 利用函數指標告訴lib要如何在root 中複製element節點，可使用事前準備好的copy函式。
3. struct\_size: 提供自己所定義的結構大小，讓lib能要建立相同size的節點

* 使用範例:

student\_t \*root2 = NULL;

btreeNode\_t \* treeRoot2 = (btreeNode\_t\*)root2;

treeRoot2 = BST\_treeCopy(treeRoot,copy,sizeof(student\_t));

------------------------------------------------------------------------

btreeNode\_t \* BST\_delete(void \* element, btreeNode\_t \* root, int

(\*compare)(void \* elementA, void \* element), void(\*copy)(void \* elementA, void \* element));

* 函式說明: 刪除在root BST中的一個節點(element)。
* 輸入要求:

1. element: 輸入所要刪除的節點。
2. btreeNode\_t \* root: 輸入要刪除節點的 BST 樹根位置。
3. copy: 利用函數指標告訴lib要如何在root 中複製element節點，可使用事前準備好的copy函式。
4. compare: 利用函數指標告訴lib要**如何**在root 中尋找要刪除的element節點。使用事前準備好的compareID函式，讓lib知道要以哪一個數值(此範例使用ID值)進行比較與收尋要刪除的節點。

* 使用範例:

treeRoot = BST\_delete(node2,treeRoot,compareID,copy);

* 回傳結果:

當該樹中有要刪除的節點時，會將其刪除後回傳刪除後該**樹根的位置**。

------------------------------------------------------------------------

bool BST\_treeEqual(btreeNode\_t \* root, btreeNode\_t \* root\_sec,int

(\*compare)(void \* elementA, void \* element));

說明:treeEqual 比較二個 BST (root和root\_sec)是否相同

* 使用範例:

Bool ans = BST\_treeEqual(treeRoot,treeRoot2,compareID);

if(ans)

printf("Equal!!\n");

else

printf("not Equal\n");