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)`
3. 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;
```

4. 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!");}
```

5. 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
```

2. btreeNode_t * root: 告訴 lib 要將 element 新增到**何處**。
3. 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:

```
inOrder search:
id = 119  eng = 50
id = 120  eng = 80
id = 123  eng = 100
id = 125  eng = 70
```

```
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");
```