

7. Binary Search Trees

msdb@korea.ac.kr

Instruction

Make a zip file named “**studentid**” that includes one folder and source codes:

Binary search tree ADT

- ▶ `binarySearchTree.h`
- ▶ `binarySearchTree.c`
- ▶ `main.c`

Make sure your codes can be properly compiled



`binarySearch
TreeADT.h`



`binarySearch
TreeADT.c`



`main.c`



`binarySearch
ADT`



`2019000000
.zip`



Submit this!

Binary Search Tree ADT

Implement Binary search tree ADT.

bstCreate - create a binary search tree (provided)

bstDestroy - destroy a binary search tree (provided)

bstInsert - insert an element in BST

- ▶ **new < node** : insert in a left node.
- ▶ **new >= node** : insert in a right node.

bstDelete - delete an element in BST

bstSearch - search tree node containing requested key

Binary Search Tree ADT - Type Definitions

```
typedef struct treeNode
{
    int data;
    struct treeNode* left;
    struct treeNode* right;
} TREE_NODE;
```

```
typedef struct
{
    int count;
    TREE_NODE* root;
} BST_TREE;
```

Binary Search Tree ADT - Functions

```
BST_TREE* bstCreate();  
void bstDestroy(BST_TREE* tree);  
static void _bstDestroy(TREE_NODE* root);  
  
bool bstInsert(BST_TREE* tree, int data);  
bool bstDelete(BST_TREE* tree, int key);  
TREE_NODE* bstSearch(BST_TREE* tree, int key);  
  
bool bstEmpty(BST_TREE* tree);  
int bstCount(BST_TREE* tree);
```

Binary Search Tree ADT - Main

```
#include <stdio.h>
#include <stdbool.h>
#include <stdlib.h>
#include "binarySearchTreeADT.h"

int main()
{
    int data[] = { 14, 23, 7, 10, 33, 56, 80, 66, 70 };
    BST_TREE* bstTree = bstCreate();

    for (int i = 0; i < sizeof(data) / sizeof(int);
        i++)
    {
        bstInsert(bstTree, data[i]);
    }

    int delKey[] = { 33, 7, 14 };
    for (int i = 0; i < sizeof(delKey) / sizeof(int);
        i++)
    {
        bstDelete(bstTree, delKey[i]);
    }

    int retrieveKey[] = { 14, 23, 7 };
```

```
    for (int i = 0; i < sizeof(retrieveKey) /
        sizeof(int); i++)
    {
        TREE_NODE* node = bstSearch(bstTree,
            retrieveKey[i]);
        if (node)
        {
            printf("Key: %d, Founded: %d\n",
                retrieveKey[i], node->data);
        }
        else
        {
            printf("No data for key: %d\n",
                retrieveKey[i]);
        }
    }

    bstDestroy(bstTree);

    return 0;
```