

7. Binary Search Trees

msdb@korea.ac.kr



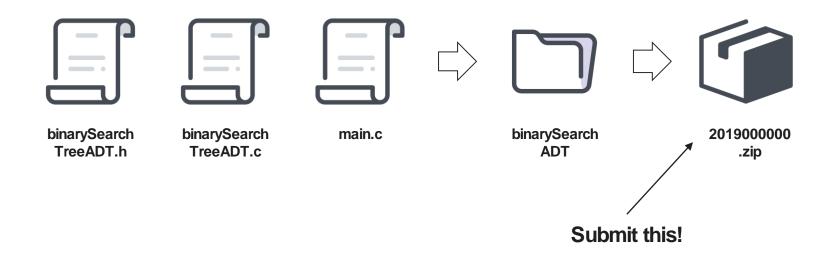
Instruction

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



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.</p>
- 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>
                                                                 for (int i = 0; i < sizeof(retreiveKey) /</pre>
                                                                 sizeof(int); i++)
#include <stdbool.h>
                                                                 {
#include <stdlib.h>
                                                                     TREE_NODE* node = bstSearch(bstTree,
#include "binarySearchTreeADT.h"
                                                                      retreiveKey[i]);
                                                                      if (node)
int main()
                                                                      {
{
                                                                          printf("Key: %d, Founded: %d\n",
     int data[] = { 14, 23, 7, 10, 33, 56, 80, 66, 70 };
                                                                          retreiveKey[i], node->data);
     BST_TREE* bstTree = bstCreate();
                                                                      else
     for (int i = 0; i < sizeof(data) / sizeof(int);</pre>
     i++)
                                                                          printf("No data for key: %d\n",
     {
                                                                          retreiveKey[i]);
          bstInsert(bstTree, data[i]);
     }
                                                                 }
     int delKey[] = { 33, 7, 14 };
                                                                 bstDestroy(bstTree);
     for (int i = 0; i < sizeof(delKey) / sizeof(int);</pre>
     i++)
                                                                 return 0;
     {
          bstDelete(bstTree, delKey[i]);
     }
     int retreiveKey[] = { 14, 23, 7 };
```