**CS 4223-01 Homework 01**
Fall 2019

**Due:** August 29 at class time

**Assignment**

Write a C program to implement and demonstrate a binary search tree of integers. The program will prompt the user to enter a list of distinct integers in arbitrary order. Use zero as a sentinel to terminate the user's entries. Insert each of the user's numbers (other than the sentinel) into a binary search tree. After the data entry is completed, do an in-order traversal of the tree to print its content on the screen. The output should be a list of the user's entries in increasing numerical order.

You must implement the binary search tree yourself. Do not use a library implementation of the tree. Construct your tree with dynamically allocated nodes connected by pointers. Your program must work correctly for an arbitrarily large number of data entries.

**Sample Screen**

```
Entry: 34
Entry: 87
Entry: 15
Entry: 112
Entry: 6
Entry: 0

Content: 6
Content: 15
Content: 34
Content: 87
Content: 112
```

**Instructions for Turning in Your Program**

Submit your C source code to this assignment on Canvas by the beginning of class on the due date.

```
//-------------------------------------------------------------
// Author ------------ Cecilia Y. Sui
// Course ------------ Compiler Construction
// Instructor --------- Dr. Crawley
// Assignment --------- Binary Search Tree Implementation in C
// Data of Submission - August 27, 2019
//-------------------------------------------------------------

#include <stddef.h>
#include<stdio.h>
#include<stdlib.h>

//-------------------------------------------------------------
// Construction of Node
//-------------------------------------------------------------
struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};

//-------------------------------------------------------------
// function to insert an element into the BST
//-------------------------------------------------------------
struct Node* insert(struct Node* root, int data){
    if (root == NULL){
        root = malloc(sizeof(struct Node));
        root->data = data;
        root->left = root->right = NULL;
    }
    else if (data < root->data) {
        root->left = insert(root->left, data);
    }
    else if (data > root->data){
        root->right = insert(root->right, data);
    }
```

```
    return root;
};

//-----------------------------------------------------------------
// inorder traversal of BST
//-----------------------------------------------------------------
void inorder(struct Node* root){
    if (root != NULL){
        inorder(root->left);
        printf("Content: %d \n", root->data);
        inorder(root->right);
    };
};

//-----------------------------------------------------------------
// main function
//-----------------------------------------------------------------
int main(){
    struct Node* root;
    int entry;
    root = NULL;
    printf("Entry: ");
    scanf("%d", &entry);
    while (entry != 0){
        root = insert(root, entry);
        printf("Entry: ");
        scanf("%d", &entry);
    }
    printf("\n");
    inorder(root);
    return 0;
};
```