# Assignment 4

Data Structures

Binary Search Trees

## Submission Notes

This assignment requires one cpp file submission. Do not upload a code snippet. Do not submit any zip or compressed files, binary files, or any other generated files. Do not put the code inside the PDF file. Submission of unnecessary files or binary files in addition to source files will make you lose the points of the assignment. The code must have the following:
a. Use consistent indentation through your code.
b. Use consistent braces style through your code.
c. File-level comments, these comments will include the file name, your name, course number, and date of last modification.
d. Function level comments include the function's description, the function's parameters, and the function's return value. These comments will appear before each of the functions, not the prototypes.
e. In function comments, these comments will include a description of the atomic functionalities within the bodies of the functions.
f. We evaluate your code using the following C++ online compiler. You will not get points if your assignment does not compile with this.

`https://www.onlinegdb.com/online_c++_compiler`

## BST Functions (100 points)

Download the BST.cpp file and complete the following functions. After completing these functions, Do not modify the rest of the codes in the file.

**a. Node* insert(Node* root, Node* element);** (15 points)

This function gets a node and the root of the Binary Search Tree and inserts the node into the BST. Then, it returns the root after insertion.

Note that we saw the implementation of this function in the class and you can use that implementation (no need to modify it). So, everyone should be able to complete this function.

**b. void inorder(Node* root);** (15 points)

This function prints the inorder traversal of the tree pointed to by root.

Note that we saw the implementation of this function in the class and you can use that implementation (no need to modify it). So, everyone should be able to complete this function.

**c. int numNodes(Node* root);** (30 points)

This function gets the root of the tree and returns the number of nodes in the tree.

**Hint:** You may implement this function recursively.

**d. int findKthSmallest(Node* root, int k);** (40 points)

This function gets the root of a BST tree and an integer k. it returns the kth smallest value in the BST pointed to by root.
Assume that k is in between 1 and the number of nodes in the tree.

**Hints:**
- You may implement this function recursively
- You may call the numNodes function implemented in the previous part inside this function.

**Example** for findKthSmallest(Node* root, int k):

Assume that you have the following inorder traversal of your BST:

2 5 16 27 38 90

If the user choose option 4 from the menu, the program asks "which ranked item would you like to find?"
Now the user should enter a value which is the input of the function (int k). Let's assume that the user enters 3. The program should returns the 3rd smallest element in the tree which is 16.