# CSE 331 - Project 3
# Binary Search Tree

TA: Sorrachai Yingchareonthawornchai
yingchar@msu.edu
Due Date: 11:59 pm Oct 10, 2014

## 1 Project Description

In this project, you will complete implementation of the binary search tree and its operations. Your binary search tree will be used to store and manipulate data from given text file.
You have been given code to implement a program that will load and manipulate the data from the numbers.txt file. The main program stores the numbers in a vector, and then builds a binary search tree based on the vector.

Your job is to complete the implementation of the following methods in BinarySearchTree class::

1. Insert: Inserts an item into the tree. No need to balance the tree. In fact, dont balance your tree because this would generate inconsistent result. This should be $O(h)$ where $h$ is height of the tree.

2. InorderTraverse: Traverses the tree in order. In addition, there is another parameter: vector. The vector is expected to have all elements in the tree by the same order as visited by each recursion. This should be $O(n)$.

3. DeleteNode: Deletes an element of the tree. This should be $O(h)$.

4. Filter: Deletes all nodes that **do not** have the desired property keeping only ones with the property. This should be $O(n)$. Provide an memory efficient implementation, i.e., $O(1)$ space.

5. ForEachInerval: perform some function to each node of the tree in range of interval $[a, b]$ where $a, b$ are parameters of the method. Provide an memory efficient implementation, i.e., $O(1)$ space. The running time should be equal to number of elements satisfying the interval.

6. VerifyBinraySearchTree: returns true if the tree has properties of binary search tree, e.g., value of left child node is less than that of its parent. Return false otherwise. This should be $O(n)$. Provide an memory efficient implementation, i.e., $O(1)$ space.

I recommend that you implement all of your functions in the private space, and then place calls to private methods in the public methods. This will reduce the amount of code you write when overloading the const methods.
You may not use anything from the STL for this project. You may use IO for debugging, but be sure to remove it from your completed project. You will be marked down for any debugging IO left in your project. As always you may not change the signatures of the public methods, or the way in which the tree class interacts with the rest of the program. You may add any private member variables or methods to BinarySearchTree class to help you complete the project.

## 2 Input Test Cases

The program will take one input which is a file containing integers line-by-line. Your program will then read file and store integers in a vector.
Your program will build a binary search tree by inserting elements from the vector. Then, your program will traverse the tree in order and recording values according to order visited into another vector. There will be manipulation of the tree using filter, forEachInterval, delete method. As usual, main.cpp does the job for you. You have to complete implementation of BinarySearchTree class. Note: use the CSE black server as a running machine. Other test cases will be executed after the due date for grading.

## 3 Project Deliverables

The following files must be submitted via Handin no later than 11:59 pm Oct 10, 2014.

1. BinarySearchTree.h - contains your implementation of binary search tree.
2. project3.pdf - file containing your answers to written questions.

## 4 Written Questions

1. Given the following set of numbers {20,10,7,13,5,22,23,54,97,33,45,21,40}
   (a) Determine the order of insertions with this set of numbers that will result in a perfectly balanced BST.
   (b) Show the results of a preorder traversal of this tree.
   (c) Delete the root, make a diagram of the resulting tree.
2. Tree Sort: we can sort a set of $n$ numbers by the followings. First, build a binary search tree by inserting the numbers one by one. Next, print the numbers by an inorder traversal. What are the worst-case and best-case running time for Tree Sort algorithm? Clearly explain why?