# COMP3512 - Assignment 3

Due: Sunday December 10, 2017 @ 11:59 PM

This is an exercise that you need to do on a computer. You'll need to commit and push your code to your GitLab repo, and submit for automated marking via Slack.

For this assignment, you will need to implement a data structure: Binary Search Tree (BST) with C++ smart pointers: shared\_ptr, weak\_ptr and unique\_ptr.

**BST is a Tree with special rules**:

1. Each node has at most 2 children, usually denoted as left child and right child.
2. Key in each node must be greater than or equal to any key stored in the left sub-tree.
3. Key in each node must be less than to any key stored in the right sub-tree.

You can find more about BST here: <https://en.wikipedia.org/wiki/Binary_search_tree>.

## 1. Project Setup

1. Open Assignment3.sln in Visual Studio 2017
2. Add TreeNode.h file to your project. (refer to Lab 1 if you don't know how)
3. Add the following content in the header file.

|  |
| --- |
| #pragma once  #include <memory>  namespace assignment3  {  template<typename T>  class TreeNode  {  public:  TreeNode(std::unique\_ptr<T> data);  TreeNode(std::shared\_ptr<TreeNode<T>> parent, std::unique\_ptr<T> data);  std::unique\_ptr<T> Data;  std::shared\_ptr<TreeNode<T>> Left;  std::shared\_ptr<TreeNode<T>> Right;  std::weak\_ptr<TreeNode<T>> Parent;  };  } |

1. Add BinarySearchTree.h and add the following content.

|  |
| --- |
| #pragma once  #include <memory>  #include <vector>  #include <iostream>  namespace assignment3  {  template<typename T>  class TreeNode;  template<typename T>  class BinarySearchTree  {  public:  void Insert(std::unique\_ptr<T> data);  bool Search(const T& data);  bool Delete(const T& data);  const std::weak\_ptr<TreeNode<T>> GetRootNode() const;  static std::vector<T> TraverseInOrder(const std::shared\_ptr<TreeNode<T>> startNode);  private:  std::shared\_ptr<TreeNode<T>> mRoot;  // any other private members of methods can go here.  };  } |

1. Note that your code may not compile because you have empty implementation.

## 2. Implement TreeNode class

* TreeNode class represents individual node in a binary search tree. It contains a "pointer" to its left child, right child and its parent. Note that Data, Left, Right and Parent are public properties, so there is no need to make getter and setter methods.
* Implement two constructors:
  + A constructor that takes in a unique pointer to data type T
    - The unique pointer here means the node will take the ownership of data T
  + Another constructor that takes an additional shared pointer to TreeNode<T>

Make sure both constructors initialize the correct property.

## 3. Implement Insert() method of BinarySearchTree class

* The new node should always be inserted at leaf.
* Rules 2) and 3) of BST above should always be true after insertion.
* Please note that unique\_ptr is used here as well.

## 4. Implement Search() method of BinarySearchTree class

* Search method should take data T as a parameter and return true if data is found inside the tree. Else return false.

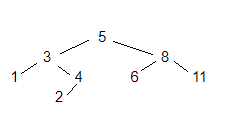
## 5. Implement Delete() method of BinarySearchTree class

* Delete method should take data T as a parameter and delete the TreeNode that stores T. If successful, return true. Else return false.
* Rules 2) and 3) of BST should still hold after deletion.

## 6. Implement TraverseInOrder() method of BinarySearchTree class

* Return a std::vector<T> that contains the result of InOrder tree traversal. If tree is empty, it should return an empty vector.

ex>



Should print out [1, 2, 3, 4, 5, 6, 8, 11]

* Please note that this is a static method that takes the starting node. One should be able to pass in any node in the BST structure and get the subtree originating from the given node.

## 7. Implement GetRoot method of BinarySearchTree class

* Returns a weak pointer to the root node of a binary search tree.

## 8. Test it

1. Add main.cpp into your project and test however you want. Below is a sample code you can use to debug.

|  |
| --- |
| #include "TreeNode.h"  #include "BinarySearchTree.h"  #include <string>  #include <iostream>  using namespace assignment3;  int main()  {  BinarySearchTree<int> tree;  tree.Insert(std::make\_unique<int>(10));  tree.Insert(std::make\_unique<int>(15));  tree.Insert(std::make\_unique<int>(5));  tree.Insert(std::make\_unique<int>(4));  tree.Insert(std::make\_unique<int>(19));  tree.Insert(std::make\_unique<int>(20));  tree.Insert(std::make\_unique<int>(17));  tree.Insert(std::make\_unique<int>(12));  std::vector<int> v = tree.TraverseInOrder(tree.GetRootNode().lock());  for (std::vector<int>::iterator it = v.begin() + 1; it != v.end(); ++it)  {  std::cout << \*it << std::endl;  }  std::cout << "--------------------------" << std::endl;  tree.Delete(6);  tree.Delete(100);  tree.Delete(15);  v = tree.TraverseInOrder(tree.GetRootNode().lock());  for (std::vector<int>::iterator it = v.begin() + 1; it != v.end(); ++it)  {  std::cout << \*it << std::endl;  }  std::cout << "--------------------------" << std::endl;  bool bSearched = tree.Search(15);  return 0;  } |

1. Compile and run it to see the expected output.

## 9. Commit, Push and Ask for a Build

You know the drill :)

BUT, please note that

* build cooldown time for assignments is 30 mins instead of 5 mins.