# HW 2. Binary Search Tree

## Environment:

- Programming Language: C++
- IDE: Atom Editor

## Requirements:

Input file (bstmap.txt or other testing file) and executable file (after compiling bst.cpp) must be at the same folder. Otherwise, the program will print out error message.

## Key Variables and Function:

- **General:**
  - class Node:

    private int value

    private Node* leftChild, rightChild

    friend class BST
  - class BST:

    private Node* root

    private stack <int> swordRoute: the route from Capoo to sword's location

    private list <int> saveRoute: the route from Capoo to meaty's location
- **Part 1:**
  - public bool BST::add(int value): If value is in it, then return false. Otherwise, add the value to BST and return true.

- public Node* BST::search(int value): If value is in BST, then return the corresponding node. Otherwise, return null.
- public void BST::del(int value): Call search function here. If node is found, replace it with the correct node's value. (First, consider the minimum in right subtree. And then the maximum in left subtree).
- public void BST::print(): It will call four kinds of traversal ways (private function).
- private void BST::levelorder(): Implement by queue. Print out the value by level-order. It will be called in BST::print function.
- private void BST::preorder(Node* current): Implement by recursive. Print out the value by pre-order. It will be called in BST::print function.
- private void BST::inorder(Node* current): Implement by recursive. Print out the value by in-order. It will be called in BST::print function.
- private void BST::postorder(Node* current): Implement by recursive. Print out the value by post-order. It will be called in BST::print function.

- **Part 2:**
  - public void BST::clearTrap(int value): It will call the private function (findTrap) (Since encapsulation, we cannot access root from public usage). Deleting the node whose value have digit same as input parameter, value.
  - public void BST::printRoute(int sword, int meaty): It will call BST::findMeaty and BST::findSword two functions. Print out the route from Capoo to Sword and from Sword to Meaty.

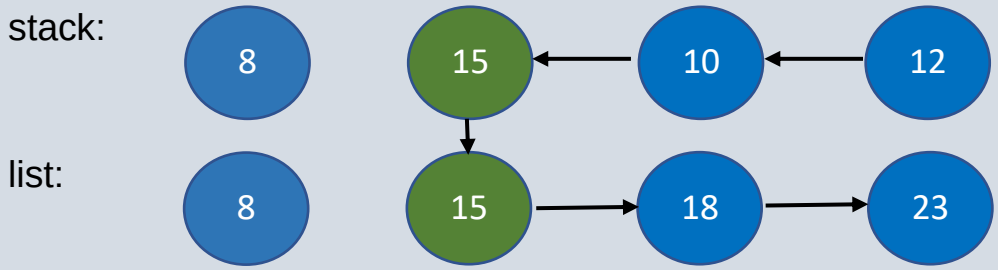
- private void BST::findMeaty(int meaty): Find meaty's location and keep tracking the route in the list (saveRoute). It will be called in BST::printRoute function.
- private void BST::findSword(int sword): Find sword's location and keep tracking the route in the stack (swordRoute). It will be called in BST::printRoute function.
- private void BST::findTrap(Node* current, int value): It will call the private function (hasValue). Find out the node and delete it whose value have digit same as input parameter, value. It will be called in BST::clearTrap function.
- bool hasValue(int number, int value): Determine whether number has digit same as value. If does return true. Otherwise, it returns false. It will be called in BST::findTrap function.

## Thought:

I just focus on part 2, finding meaty. Because part 1 is all simple operations of binary search tree, let us assume that they are all well-defined here.

First, load the map and construct it. Then, based on the trap's value, delete the corresponding node by calling the clearTrap function in post-order traverse. Next, according to sword and meaty's location, use findSword and findMeaty two functions to generate swordRoute and saveRoute, two variables, respectively. Finally, we go backward from sword position, which means we need to go back by the stack (swordRoute) and pop one out. Every time we go back, we need to check whether the node we are on is also inside the list (saveRoute) or not. If so, we then don't need to go back anymore.

Eventually, we go through the list starting at that node and we can save meaty. For example,




the shortest path: 12->10->15->18->23

## Problems:

- Because of the output spec, we need to generate the strange code that are not easy to understand.
- My binary search tree doesn't have parent pointer. Therefore, when I am deleting the node, to get the parent node become the big trouble. Hence, I need to search the tree again and find the parent. However, parent pointer only useful for deleting. I think it doesn't pays to construct the BST with parent pointer.(need more space)