Assignment 4 – Binary Search Tree

TA: Ruby (wei929098@hotmail.com)
Deadline: Dec. 7, 11:59pm

1. Implement a Binary Search Tree with linked lists (40%)

A binary search tree (BST) is a sorted tree structure where each node has two children. The left node always needs to contain a value smaller than the parent and the right node always needs to contain a value equal or greater than the parent. Exception is when the node contains no data.

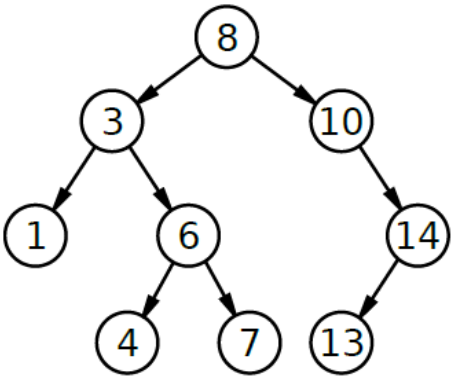

Figure 1: Example of a BST

In this assignment, you need to implement a BST structure for storing integer numbers and provide the following functions:

- **insert (i)** let the user insert a number i; error out when the number is already existing.
- **delete (i)** let the user delete a number i; error out when the number does not exist. If the node with two child nodes is deleted, choose the smallest one on the right subtree to replace the deleted node.
- **search (i)** search the tree; message out whether the number i is found or not.
- **printInfixOrder()** print the whole tree in infix order (from left to right)
- **printLevelOrder()** print the whole tree in level order (from up to down)

To help you getting started with this assignment, here is a piece of code you can use. It is the code for a node in a tree. The node can contain two child nodes, left and right.

```
struct node
{
  int data;
  struct node *left;
  struct node *right;
};
```

2. Treasure Hunter (50%)

The function of treasure hunter is very similar to Part one. In addition to implementing a BST, you need to implement a treasure hunter function to look for coins on the tree. The scenario is described as follows:

There are two treasure hunters who go into the maze represented by a binary search tree to find the coins. They represent two different numbers. According to the numbers, they follow the BST rule to go into the maze and pass the nodes. When passing the nodes, the treasure hunter collects the coins. Finally, they tally the coins. The treasure hunter with most coins wins!

In this part of assignment, you are given a map file to construct the maze. The format of file is as follows.

```
8,1
3,2
1,4
13,3
10,6
6,5
4,9
12,12
9,11
7,10
18,7
15,13
2,8
19,14
```

Insert each number sequentially into the binary search tree. Give each node a number to represent the coins. The illustration of the maze is shown in Figure 2.

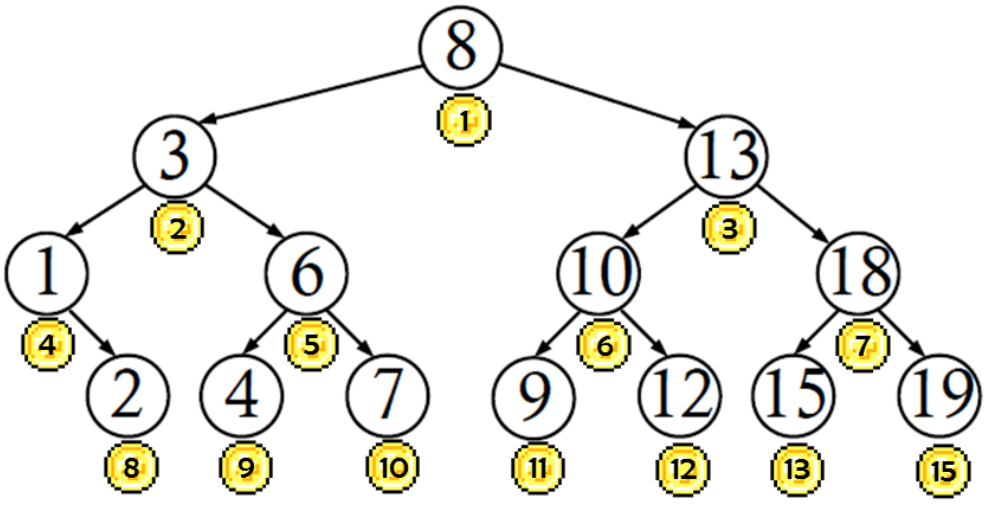

Figure 2: Example of a maze

For example, assume one of the treasure hunter represents 5, the other represents 27.

Figure 3: Example of a maze

The former's walking trajectory is 8→3→6→4. He collects 1+2+5+9=17 coins. The latter's walking trajectory is 8→13→18→19. He collects 1+3+7+14=25 coins. Finally, the #27 treasure hunter wins the game!

ATTENTION: You must output a result.txt including the two hunters' walking trajectories and the game results. Taking #5 hunter as an example:

```
#5 hunter's walking trajectory
node:8 collect 1 coins
node:3 collect 2 coins
node:6 collect 5 coins
node:4 collect 9 coins
Totally collected 17 coins
```

3. Output file, readme, comments and style (10%)

The TA(s) will mark and give points according to the following:

- 40% Part 1 source code can be compiled without any errors and the results are correct.
- 50% Part 2 source code can be compiled without any errors and the results are correct.
- 10% Readme file, code style, and comments in source code

Readme file should include your name, class ID, a brief description of the code, and other issues students think that will be helpful for the TAs to understand their homework.

4. How to submit

To submit your files electronically, enter the following command from the csie workstation:

turnin ds.hw4 [your files…]

To check the files you turnin, enter the following command from the csie workstation:

turnin –ls ds.hw4

You can see other description about turnin from following link:
https://www.cs.ccu.edu.tw/lab401/doku.php?id=turninhowto