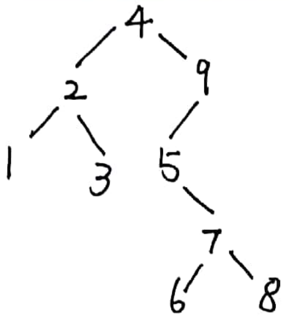
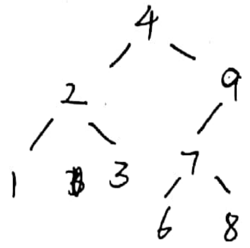
JOINT INSTITUTE
交大密西根学院

Name: Bingcheng HU
Student ID: 516021910219
Course Code: VE281
Date: Assignment 5

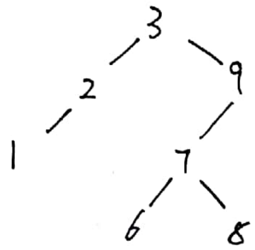
1. (a)



(b)



(c)



2. To verify a binary tree to be a BST, we must check every node. If we check every node exactly once, it will be the most efficient way.
The run time efficiency should be Θ(n) because every node is checked.

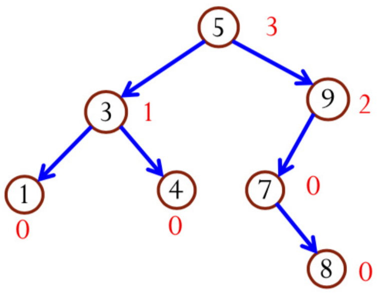
Algorithm :

```
Input : non-empty tree root
Output : is BST ?
  func BSTcheck (root)
    isBST ← true
    if root.left.~~key~~ is empty
        do nothing
    else if root.left.key > root.key
        return false.
    else
        return BSTcheck (root.left)
    end if
    if root.right is empty
    else if root.right.key < root.key
        return false.
    else
        return BSTcheck( root.right)
    end if
    return isBST
  end function.
```

```
node* getPredHelper(node* root, Key key, node* parent, bool& flag){
    if (root -> key == key)
    {
        if (root->left != NULL)
        {
            return findMax(root->left);
        }
        flag = true;
        return NULL;
    }
    if (root->key > key)
    {
        return getPredHelper(root->left, key, root, flag);
    }
    else if (root->key < key)
    {
        node* temp = getPredHelper(root->right, key, root, flag);
        if(flag){
            flag = false;
            return root;
        }
        return temp;
    }
    return NULL;
}

node* getPred(node* root, Key key){
    bool flag = false;
    return getPredHelper(root, key, NULL, flag);
}
```

Let's test this program with the following BST.



I have tested this program with `main()` below.

```
int main(int argc, char *argv[]) {
    const int array_size = 7;
    Key array[array_size] = {5,3,9,1,4,7,8};
    node *root = new node(array[0]);
    for (int i = 1; i < array_size; ++i)
    {
        insert(root, array[i]);
    }
    cout<<"depth is "<<depth(root)<<endl;
    print_tree(root);
    Key key = 1;
    for (int i = 0; i < array_size; ++i)
    {
        cout<<" Get Predecessor of ["<< array[i] <<"] is [";
        print_node(getPred(root, array[i]));
        cout<<"]"<<endl;
    }
}
```

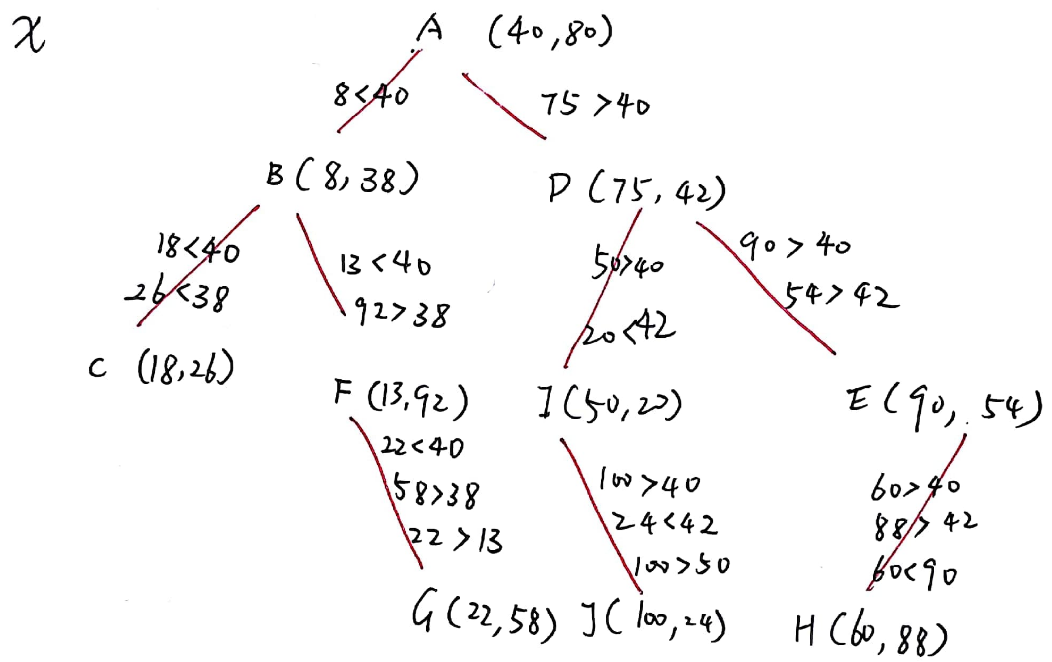
The answer is shown in the terminal

```
$ make
g++ -g -o bst BST.cpp
./bst
depth is 3
[5(3(1, 4), 9(7(N, 8), N))]
 Get Predecessor of [5] is [4]
 Get Predecessor of [3] is [1]
 Get Predecessor of [9] is [8]
 Get Predecessor of [1] is [NULL]
 Get Predecessor of [4] is [3]
 Get Predecessor of [7] is [5]
 Get Predecessor of [8] is [7]
```

Which is absolutely right!

Q4. Insert from A to J one by one.

Comparison dimension of the root is the x dimension.


x
A (40,80)
8<40
75>40
B (8,38)
D (75,42)
18<40
26<38
13<40
92>38
50>40
20<42
90>40
54>42
C (18,26)
F (13,92)
I (50,22)
E (90, 54)
22<40
58>38
22>13
100>40
24<42
100>50
60>40
88>42
60<90
G (22,58)
J (100,24)
H (60,88)