Sukkur IBA University
Department of Computer Science

DATA STRUCTURES
Lab06 – Trees

Instructor: Saif Hassan

READ IT FIRST

Prior to start solving the problems in this assignments, please give full concentration on following points.

1. WORKING – This is individual lab. If you are stuck in a problem contact your teacher, but, in mean time start doing next question (don't waste time).
2. DEADLINE – 11th March, 2022
3. SUBMISSION – This assignment needs to be submitted in a soft copy.
4. WHERE TO SUBMIT – Please visit your LMS.
5. WHAT TO SUBMIT – Submit this docx and pdf file.

KEEP IT WITH YOU!

1. Indent your code inside the classes and functions. It's a good practice!
2. It is not bad if you keep your code indented inside the loops, if and else blocks as well.
3. Comment your code, where it is necessary.
4. Read the entire question. Don't jump to the formula directly.

I, ___**Amjad Ali**___ with student ID _**191-21-0001**_

Section _**A**_hereby declare that I do understand the instructions above and follow them. This is

my own work.

Exercises

Task1 Description

Task 01: (Insertion in Binary Tree)

Binary Tree: A tree whose elements have at most 2 children is called a binary tree. Since each element in a binary tree can have only 2 children, we typically name them the left and right child.

You have been provided above the code for **Node** class, your task is to complete **BinaryTree** Class:

```
1.  class BinaryTree
2.  {
3.      // Root of Binary Tree
4.      Node root;
5.
6.      // Constructors
7.      BinaryTree(int key)
8.      {
9.          root = new Node(key);
10.     }
11.
12.     BinaryTree()
13.     {
14.         root = null;
15.     }
16.
17.     // Methods
18.     public void addData(int data) {
19.
20.         // insert elements in a tree so that left subtree of parent should contain smaller values
21.         // and right sub-tree should contain larger than its parent.
22.         // handle all possible exceptions/errors
23.     }
24.
25.     public boolean searchData(int data) {
26.
27.         // search data from Binary Tree and return true/false, check all possible conditions
28.         // handle all possible exceptions/errors
29.     }
30.
31.     public static void main(String[] args
32.
33.     {
34.         // Test the main method by creating node for different multiple nodes with children

35.     }
36. }
```

Solution:

```
1. class Node {
2.     int data;
3.     Node left;
4.     Node right;
5.
6.     Node(int data) {
7.         this.data = data;
8.         left = null;
9.         right = null;
10.     }
11.
12.     @Override
13.     public String toString() {
14.         return "Node= " + data;
15.     }
16. }
17.
18. public class BinaryTree {
19.     //Root of Binary Tree
20.     Node root;
21.
22.     // Constructors
23.
24.     BinaryTree(int key) {
25.         root = new Node(key);
26.
27.     }
28.
29.
30.     BinaryTree() {
31.         root = null;
32.
33.     }
34.
35.     // Methods
36.
37.
38.     public void addData(int data) {
39.         var node = new Node(data);
40.         if (root == null) {
41.             root = node;
42.             return;
43.         }
44.
45.         if (data == root.data) {
46.             System.out.println("Duplicate");
47.             return;
```

```
48.
49.         }
50.
51.         var current = root;
52.
53.         while (true) {
54.             // insert elements in a tree so that left subtree of
    parent should contain smaller values
55.             // and right sub-tree should contain larger than its
    parent.
56.             // handle all possible exceptions/errors
57.             if (data == current.data) {
58.                 System.out.println("Duplicate");
59.                 return;
60.             }
61.             if (current.data < data) {
62.                 if (current.right == null) {
63.                     current.right = node;
64.                     break;
65.                 }
66.
67.                 current = current.right;
68.             } else if (current.data > data) {
69.                 if (current.left == null) {
70.                     current.left = node;
71.                     break;
72.                 }
73.                 current = current.left;
74.
75.
76.             }
77.         }
78.
79.
80.     }
81.
82.
83.     public boolean searchData(int data) {
84.
85.         // search data from Binary Tree and return true/false, check
    all possible conditions
86.         // handle all possible exceptions/
87.         if (root == null) {
88.             System.out.println("Tree is Empty");
89.             return false;
90.         }
91.
92.
93.         Node current = root;
94.         while (current != null) {
```

```
95.                 if (current.data == data) {
96.                     return true;
97.                 } else if (current.data < data) {
98.                     current = current.right;
99.                 } else {
100.                    current = current.left;
101.                }
102.
103.            }
104.
105.            return false;
106.        }
107.
108.        public static void main(String[] args) {
109.            BinaryTree tree = new BinaryTree();
110.            tree.addData(10);
111.            tree.addData(5);
112.            tree.addData(8);
113.            tree.addData(23);
114.            tree.addData(15);
115.            tree.addData(6);
116.            tree.addData(7);
117.            System.out.println("Done");
118.
119.        }
120.
121.
122. }
```

Sample Input:

```
BinaryTree tree = new BinaryTree();
tree.addData(10);
tree.addData(5);
tree.addData(8);
tree.addData(23);
tree.addData(15);
tree.addData(6);
tree.addData(7);
System.out.println("Done");
```

Sample Output

```
"C:\Program Files\Java\jdk-16.0.2\bin\java.exe" "-javaagent:C:\Pro
 .jar=60256:C:\Program Files\JetBrains\IntelliJ IDEA Community Edi
 C:\Users\he\IdeaProjects\Stack\out\production\Stack BinaryTree
Done

Process finished with exit code 0
```

Task2 Description

Task 02: (Tree Traversal)

Modify Task 01 and design following methods to access tree elements in different ways

a) Tree: Preorder Traversal
b) Tree: Postorder Traversal
c) Tree: Inorder Traversal
d) Tree: Height of a Binary Tree

Solution:

(preOrder Treversal)

```
1. public static void preOrder(Node root) {
2.         // NLR
3.         if (root == null)
4.             return;
5.         System.out.print(root.data + ", ");
6.         preOrder(root.left);
7.         preOrder(root.right);
8.     }
```

(postOrder Treversal)

```
1. public static void postOrder(Node root) {
2.         // RLN
3.         if (root == null)
4.             return;
5.
6.         postOrder(root.right);
7.         postOrder(root.left);
8.         System.out.print(root.data + ", ");
9.     }
```

(inOrder Treversal)

```
1. public static void inOrder(Node root) {
2.
3.         //LNR
4.         if (root == null)
5.             return;
6.         inOrder(root.left);
7.         System.out.print(root.data + ", ");
```

```
8.          inOrder(root.right);
9.      }
```

(Hieght of tree)

```
1. public static int hieghtOfTree(Node root) {
2.          if (root == null)
3.              return 0;
4.
5.          return Math.max(hieghtOfTree(root.right), hieghtOfTree(root.left)) +
   1;
6.      }
```

Sample Input:

```
System.out.println("Hieght = "+tree.hieghtOfTree(tree.root));

System.out.print("inOrder: ");
tree.inOrder(tree.root);

System.out.println();

System.out.print("preOrder: ");
tree.preOrder(tree.root);

System.out.println();

System.out.print("postOrder: ");
tree.postOrder(tree.root);

System.out.println("\n Done");
```

Sample Output

```
Hieght = 5
inOrder: 5  6  7  8  10  15  23
preOrder: 10  5  8  6  7  23  15
postOrder: 7  6  8  5  15  23  10
 Done
```

Task3 Description

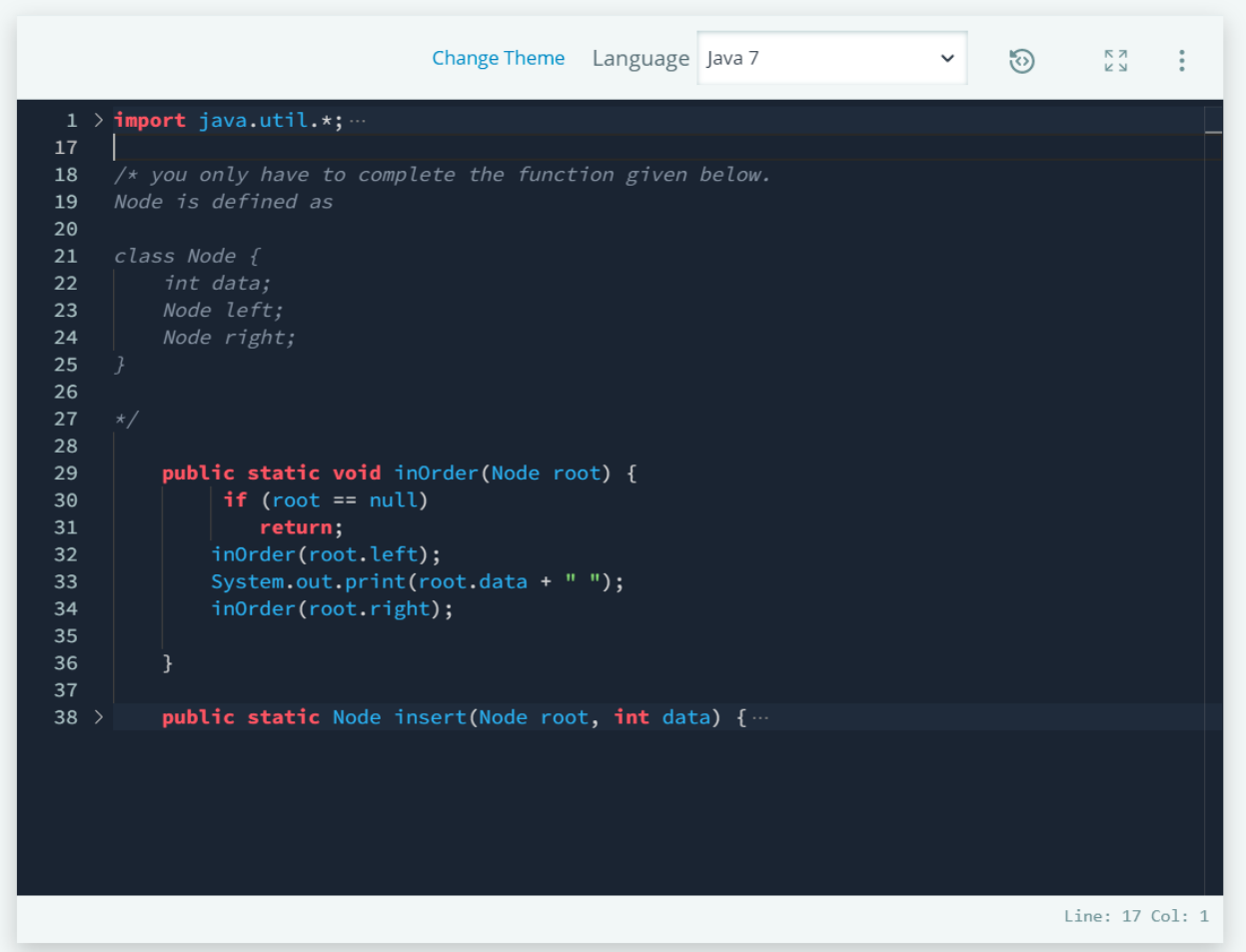
Task 03: (Join contest on Hackerrank)

A contest has been created on **hackerrank** website. First signup for the contest and start doing following assigned.

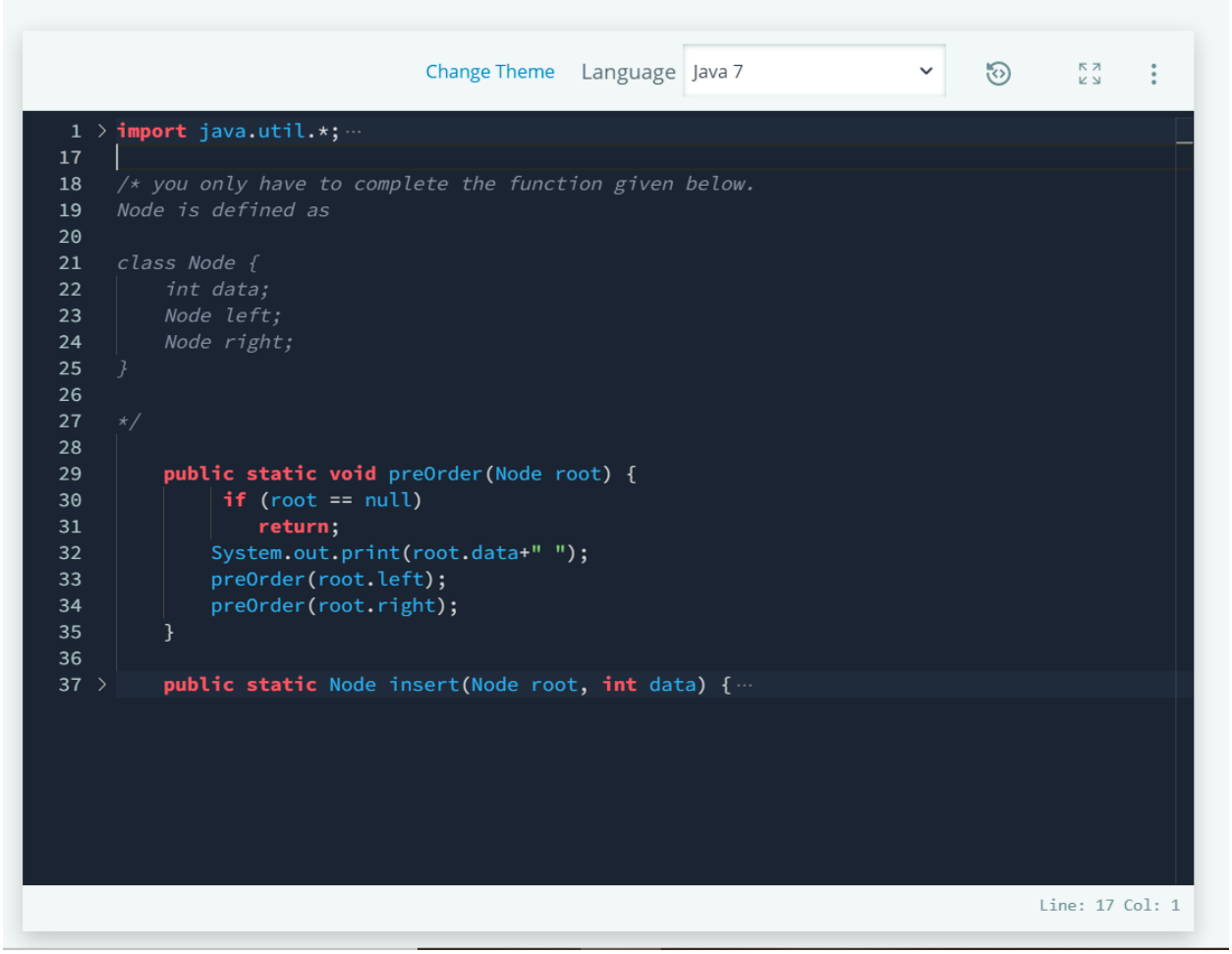
a) Tree: Preorder Traversal
b) Tree: Postorder Traversal
c) Tree: Inorder Traversal
d) Tree: Height of a Binary Tree

Sample Input:

inOrder Traversal



preOrder Traversal



postOrder Traversal

Change Theme Language Java 7

```
1  > import java.util.*; ···
17
18   /* you only have to complete the function given below.
19   Node is defined as
20
21   class Node {
22       int data;
23       Node left;
24       Node right;
25   }
26
27   */
28
29       public static void postOrder(Node root) {
30           if (root == null)
31               return;
32
33
34           postOrder(root.left);
35           postOrder(root.right);
36           System.out.print(root.data + " ");
37
38       }
39
40  >    public static Node insert(Node root, int data) { ···
```

Line: 17 Col: 1

Sample Output> nOrder Traversal

Congratulations

You solved this challenge. Would you like to challenge your friends?

Next Challenge

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Compiler Message

Success

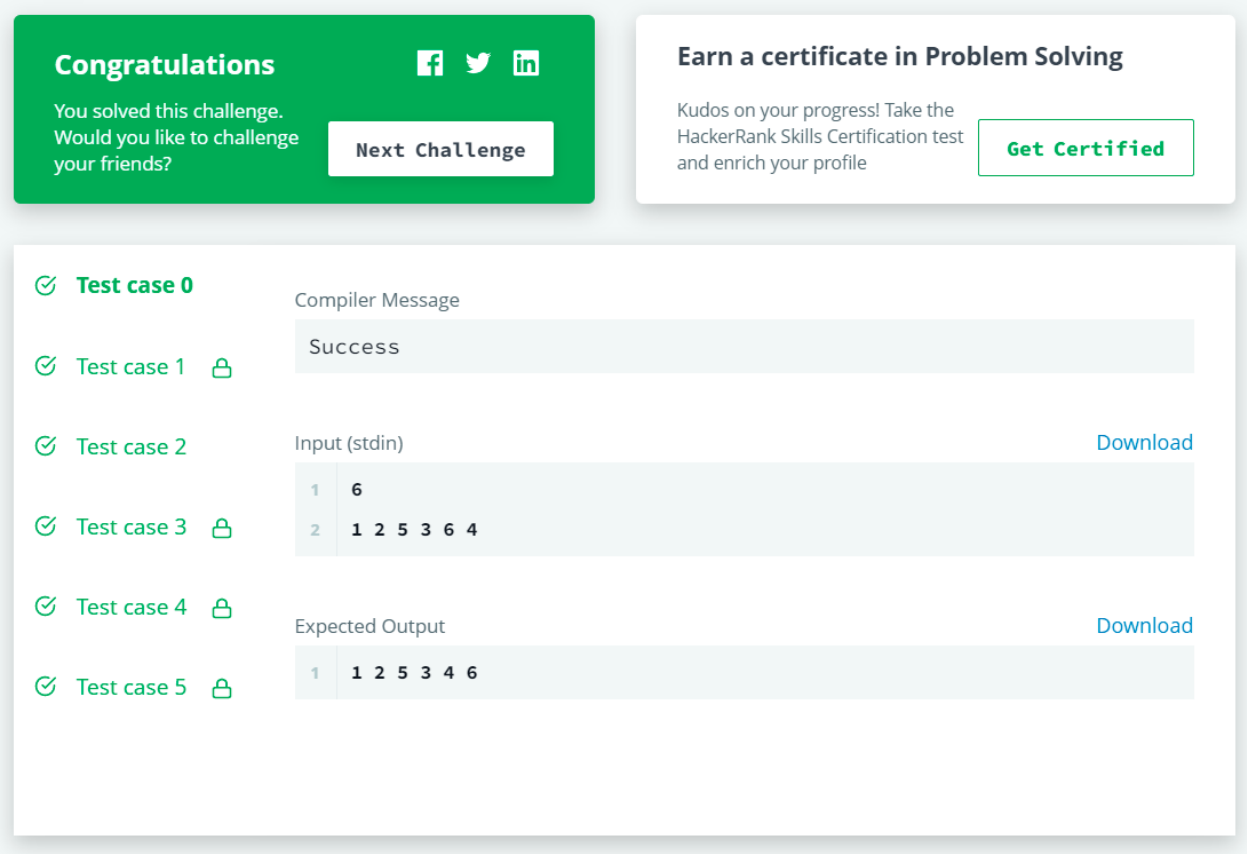
Input (stdin) Download

```
1  6
2  1 2 5 3 6 4
```

Expected Output Download

```
1  1 2 3 4 5 6
```

preOrder Traversal



postOrder Traversal