|  |  |  |
| --- | --- | --- |
| Sukkur_IBA_New_Logo | **Sukkur IBA University**  **Department of Computer Science** | **C:\Users\Saif Hassan\Downloads\CS logo (3).jpg** |

**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.

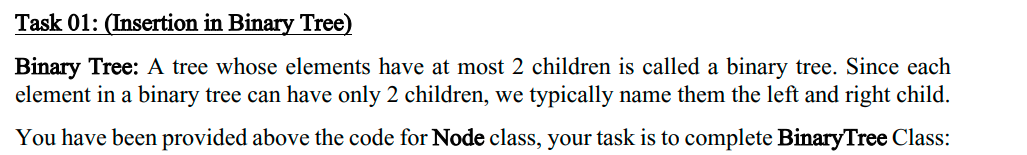
I, \_\_\_\_**Amjad Ali**\_\_\_ with student ID \_**191-21-0001**\_

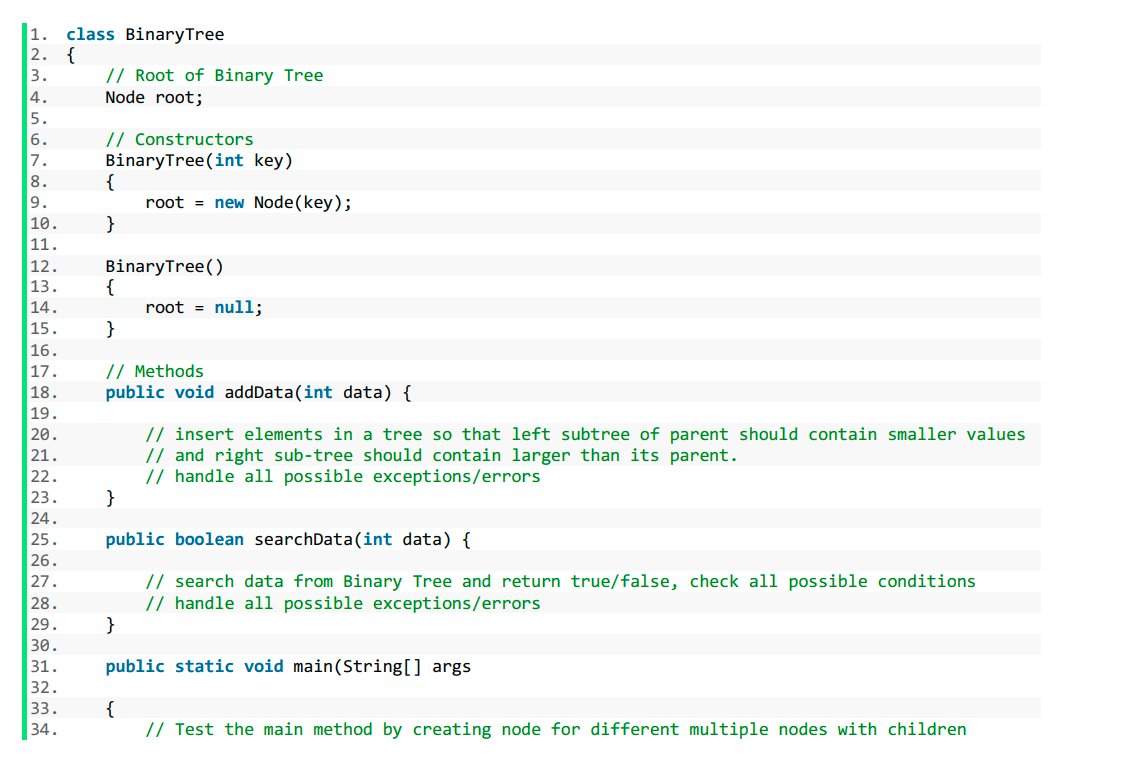
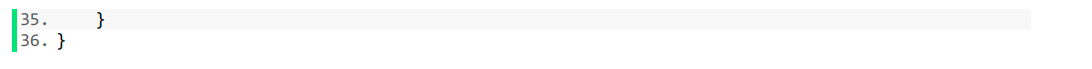
Section \_**A**\_hereby declare that I do understand the instructions above and follow them. This is

my own work.

**Exercises**

**Task1 Description**





Solution:

1. class Node {
2. int data;
3. Node left;
4. Node right;
6. Node(int data) {
7. this.data = data;
8. left = null;
9. right = null;
10. }
12. @Override
13. public String toString() {
14. return "Node= " + data;
15. }
16. }
18. public class BinaryTree {
19. //Root of Binary Tree
20. Node root;
22. // Constructors
24. BinaryTree(int key) {
25. root = new Node(key);
27. }

30. BinaryTree() {
31. root = null;
33. }
35. // Methods

38. public void addData(int data) {
39. var node = new Node(data);
40. if (root == null) {
41. root = node;
42. return;
43. }
45. if (data == root.data) {
46. System.out.println("Duplicate");
47. return;
49. }
51. var current = root;
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. }
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;

76. }
77. }

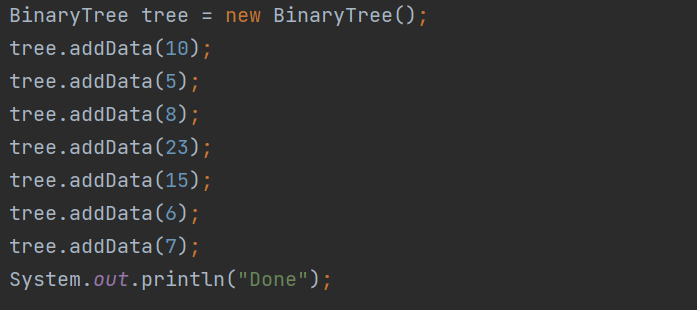
80. }

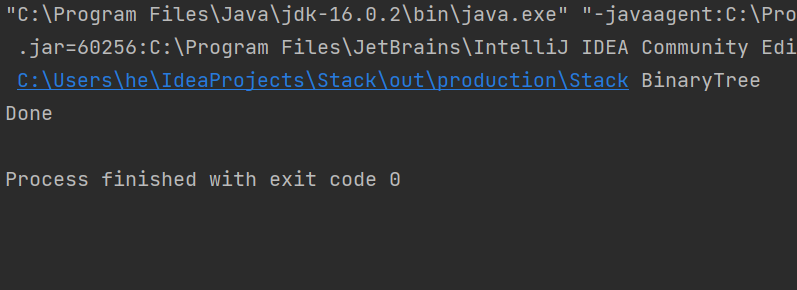
83. public boolean searchData(int data) {
85. // search data from Binary Tree and return true/false, check all possible conditions
86. // handle all possible exceptions/
87. if (root == null) {
88. System.out.println("Tree is Empty");
89. return false;
90. }

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. }
103. }
105. return false;
106. }
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");
119. }

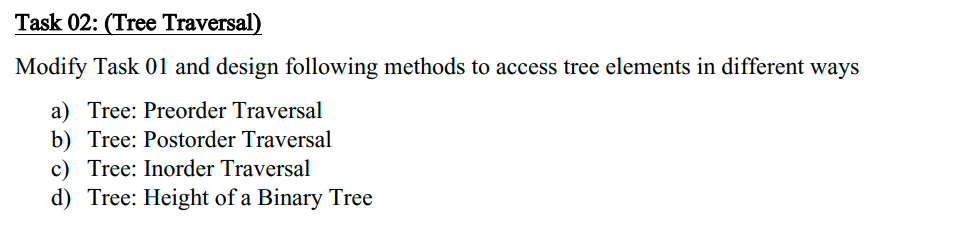
122. }

**Sample Input:**

**Sample Output**



**Task2 Description**



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;
6. postOrder(root.right);
7. postOrder(root.left);
8. System.out.print(root.data + ", ");
9. }

**(inOrder Treversal)**

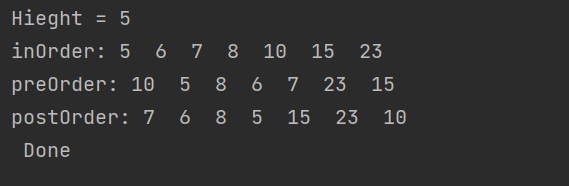
1. public static void inOrder(Node root) {
3. //LNR
4. if (root == null)
5. return;
6. inOrder(root.left);
7. System.out.print(root.data + ", ");
8. inOrder(root.right);
9. }

**(Hieght of tree)**

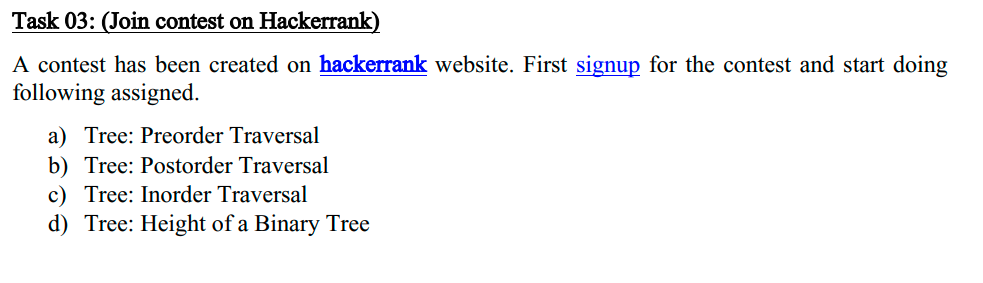
1. public static int hieghtOfTree(Node root) {
2. if (root == null)
3. return 0;
5. return Math.max(hieghtOfTree(root.right), hieghtOfTree(root.left)) + 1;
6. }

**Sample Input:**

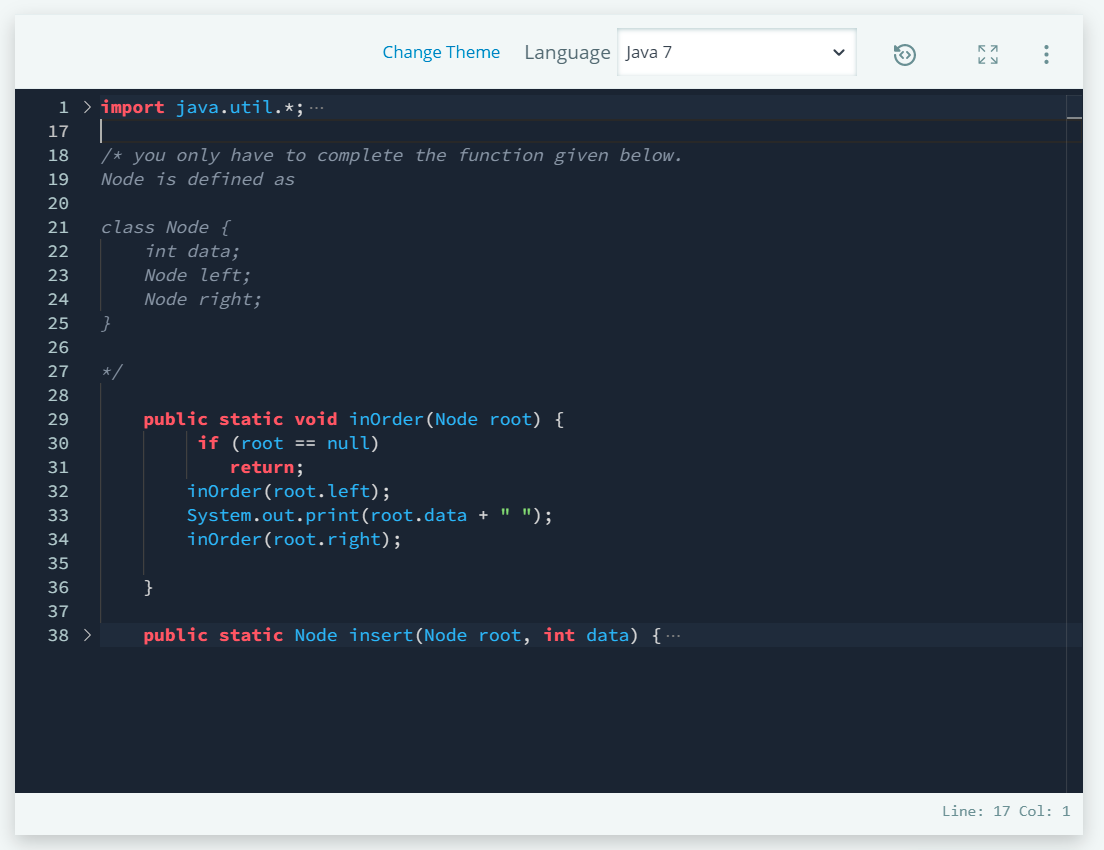
**Sample Output**

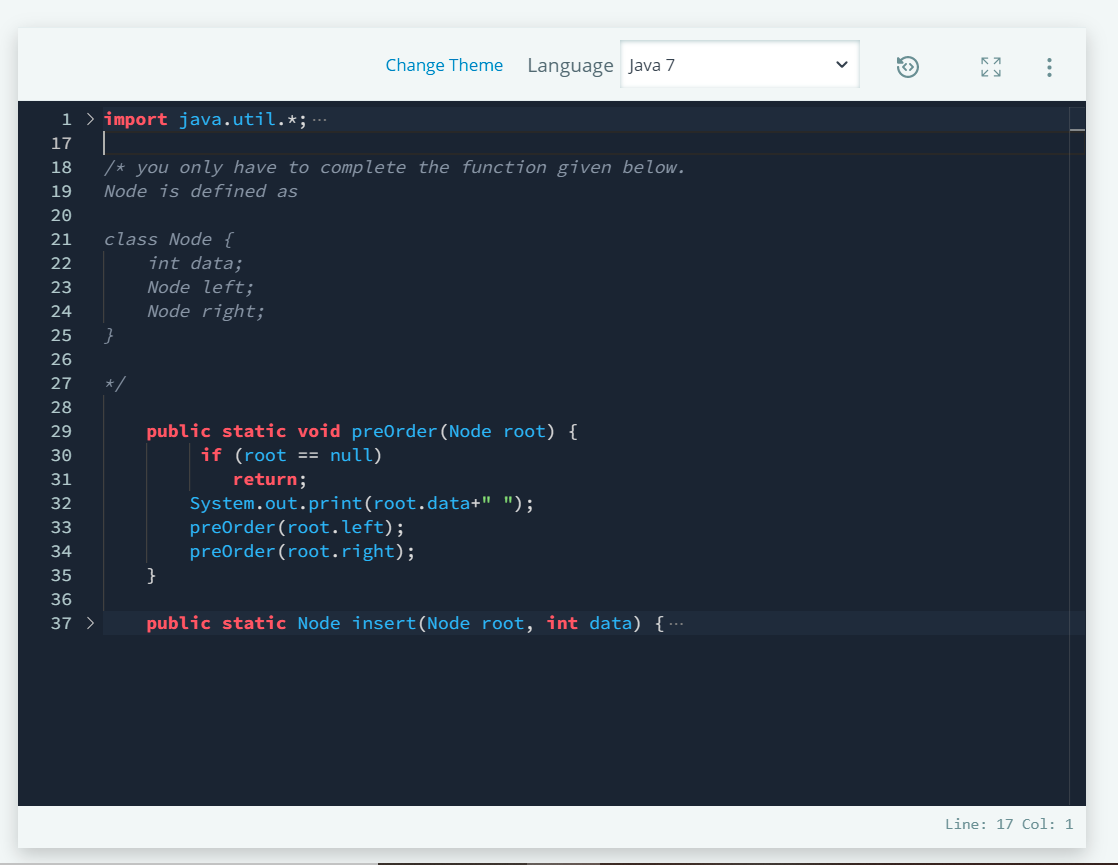


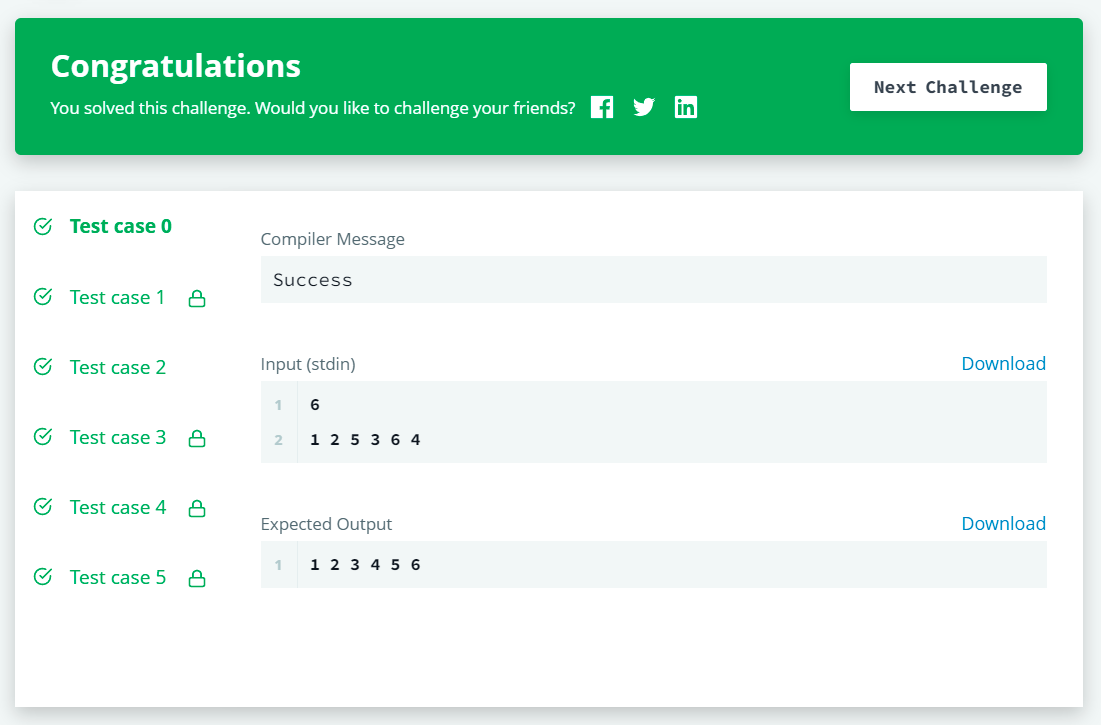
**Task3 Description**

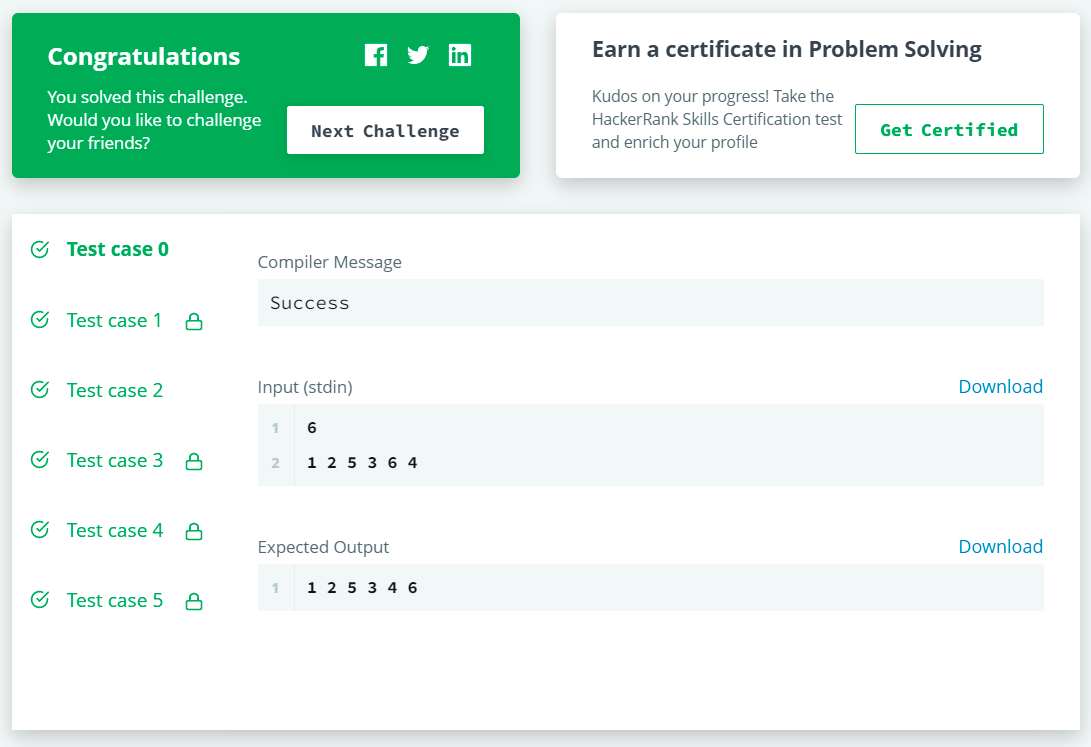


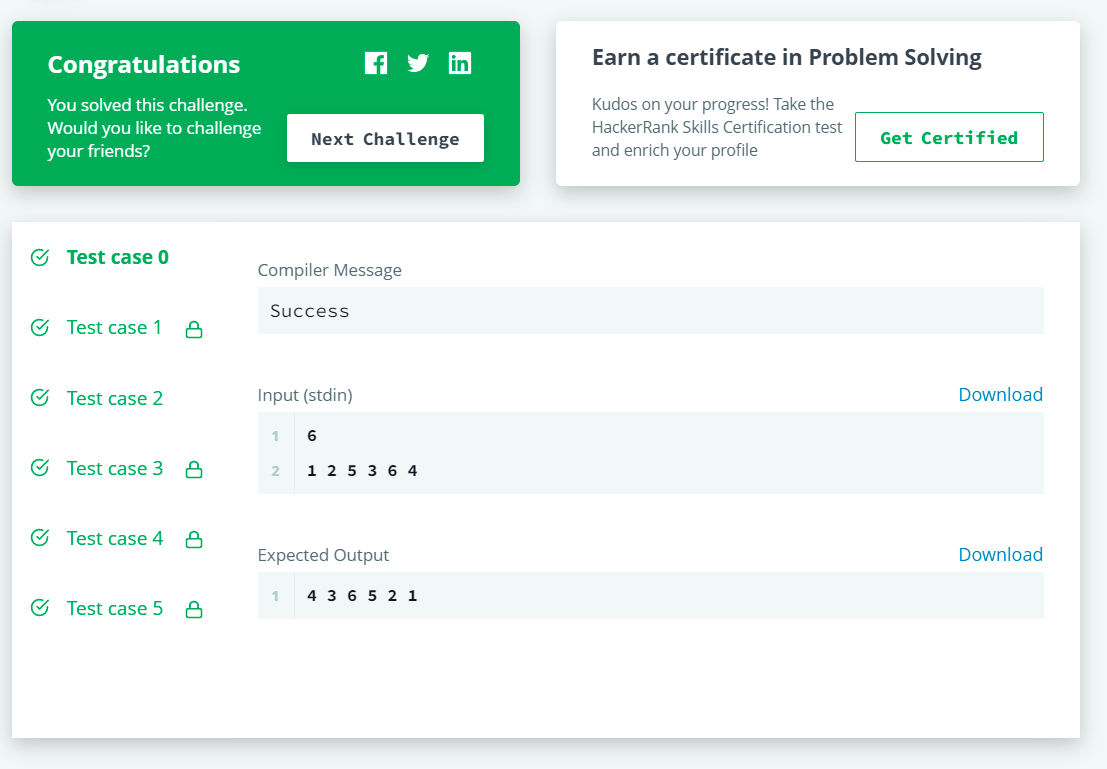
**Sample Input:**

**inOrder Traversal**

**preOrder TraversalpostOrder Traversal**

**Sample Output> nOrder Traversal**

**preOrder Traversal**

**postOrder Traversal**