New University logo


|  |  |  |  |
| --- | --- | --- | --- |
| Academic Year | 2023 | | |
| Semester | Fall | Winter | Summer |
| Course Code - Name | CSCI 2010U – Data Structures | | |
| Instructor | Dr. Razi Iqbal | | |
| Assessment | Exercise 8 |  | |

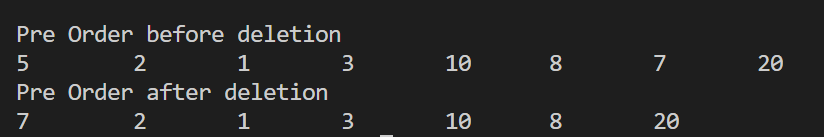
**Question (Intermediate, Advanced)**

This exercise tests your knowledge of Binary Search Trees in Java.

You are required to write Java code for deletion of nodes from a Binary Search Tree. Deletion should work if:

* Tree is null (root = null)
* Node to be deleted has no child
* Node to be deleted has 1 child
* Node to be deleted has 2 children
* Node to be deleted is root
  + This is a special case in which you should replace the root with the second highest element after the root. You can do that by going right from the root and then finding the left most node.

Below is the screenshot of the expected output of this program when root (5) is deleted:



Try to run the program using commands in terminal to get more practice.

**Question (Solution)**

**BinarySearchTree.java**

class Node

**{**

int value**;**

Node left**;**

Node right**;**

Node **(**int value**)**

**{**

**this.**value **=** value**;**

**}**

**}**

public class BinarySearchTree **{**

Node root**;**

void deletion**(**int value**)**

**{**

root **=** delete**(**root**,** value**);**

**}**

Node delete**(**Node node**,** int value**)**

**{**

**if** **(**node **==** **null)**

**{**

**return** node**;**

**}**

**if** **(**value **<** node**.**value**)**

**{**

node**.**left **=** delete**(**node**.**left**,** value**);**

**}**

**else** **if** **(**value **>** node**.**value**)**

**{**

node**.**right **=** delete**(**node**.**right**,** value**);**

**}**

**else**

**{**

**if** **(**node**.**left **==** **null)**

**{**

**return** node**.**right**;**

**}**

**else** **if** **(**node**.**right **==** **null)**

**{**

**return** node**.**left**;**

**}**

node**.**value **=** minValue**(**node**.**right**);**

node**.**right **=** delete**(**node**.**right**,** node**.**value**);**

**}**

**return** node**;**

**}**

int minValue**(**Node node**)** **{**

int minValue **=** node**.**value**;**

**while** **(**node**.**left **!=** **null)** **{**

minValue **=** node**.**left**.**value**;**

node **=** node**.**left**;**

**}**

**return** minValue**;**

**}**

void preOrder**(**Node node**)**

**{**

**if** **(**node **==** **null)**

**{**

**return;**

**}**

System**.**out**.**print**(**node**.**value **+** "\t "**);**

preOrder**(**node**.**left**);**

preOrder**(**node**.**right**);**

**}**

**}**

**Driver.java**

public class Driver **{**

public static void main**(**String**[]** args**)** **{**

BinarySearchTree tree **=** **new** BinarySearchTree**();**

tree**.**root **=** **new** Node**(**5**);**

tree**.**root**.**left **=** **new** Node**(**2**);**

tree**.**root**.**right **=** **new** Node**(**10**);**

tree**.**root**.**left**.**left **=** **new** Node**(**1**);**

tree**.**root**.**left**.**right **=** **new** Node**(**3**);**

tree**.**root**.**right**.**left **=** **new** Node**(**8**);**

tree**.**root**.**right**.**right **=** **new** Node**(**20**);**

tree**.**root**.**right**.**left**.**left **=** **new** Node**(**7**);**

System**.**out**.**println**(**"\nPre Order before deletion"**);**

tree**.**preOrder**(**tree**.**root**);**

System**.**out**.**println**(**"\nPre Order after deletion"**);**

tree**.**deletion**(**5**);**

tree**.**preOrder**(**tree**.**root**);**

**}**

**}**