Competitive Programming Lab - 2

Academic year: 2020-2021 **Semester:** Long Sem

Faculty Name: Dr. Ajith Jublison sir **Date:** 05 / 6/2022

Student name: Taran Mamidala **Reg. no.:** 19BCE7346

Q1.) Convert a binary tree to a tree with single child except the first level

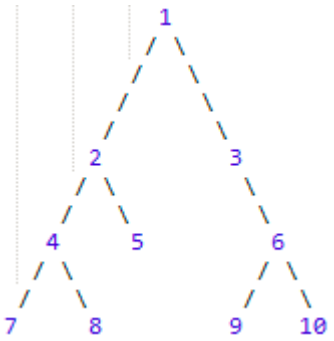
CODE:

```
class Node {
  int data;
  Node left, right;

  Node(int item) {
      data = item;
      right = left = null;
  }
}

public class BinaryTreeConversion {
  Node root;
  Node head;
  Node prev;
  public Node GettingLeafNodes(Node root) {
      if (root == null)
      return null;
      if (root.left == null && root.right == null) {
      if (head == null) {
      head = root;
      prev = root;
      } else {
      prev.right = root;
      root.left = prev;
      prev = root;
      }
      return null;
      }
      root.left = GettingLeafNodes(root.left);
```

```
        root.right = GettingLeafNodes(root.right);
        return root;
    }

    void inorder(Node node) {
        if (node == null)
        return;
        inorder(node.left);
        System.out.print(node.data + " ");
        inorder(node.right);
    }

    public void twoElementsSearch(Node head) {
        Node last = null;
        while (head != null) {
        System.out.print(head.data + " ");
        last = head;
        head = head.right;
        }
    }

    public static void main(String args[]) {
        BinaryTreeConversion tree = new BinaryTreeConversion();
        tree.root = new Node(1);
        tree.root.left = new Node(2);
        tree.root.right = new Node(3);
        tree.root.left.left = new Node(4);
        tree.root.left.right = new Node(5);
        tree.root.right.right = new Node(6);
        tree.root.left.left.left = new Node(7);
        tree.root.left.left.right = new Node(8);
        tree.root.right.right.left = new Node(9);
        tree.root.right.right.right = new Node(10);

        System.out.println("Taken Binary tree : ");
        tree.inorder(tree.root);

        tree.root.left.left.left.right = new Node(0);
        tree.root.right.right.right.right = new Node(0);

        tree.GettingLeafNodes(tree.root);
System.out.println("\nRemoving Leaf Nodes from Binary tree : ");
```

```
        tree.twoElementsSearch(tree.head);

        System.out.println("\nResult of tree obtained : ");
        tree.inorder(tree.root);
    }
}
```

Input Binary Tree :

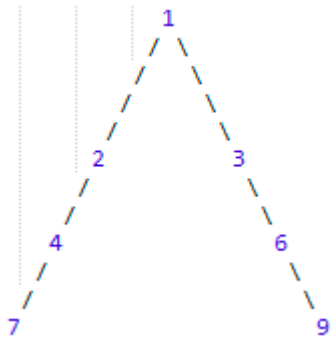


Output:

Result

CPU Time: 0.11 sec(s), Memory: 33420 kilobyte(s)

```
Taken Binary tree :
7 4 8 2 5 1 3 9 6 10
Removing single Leaf Nodes of a Binary tree :
8 5 10
Result of tree obtained :
7 4 2 1 3 9 6
```



Q2.) Print linked list except antepenultimate

CODE:

```
import java.util.*;
public class LinkedList {

  public static class ListNode < T > {

      T val;
      ListNode next;
      public ListNode() {

      }
      public ListNode(T val) {

      this.val = val;
```

```
    }
    public ListNode(T val, ListNode next) {

    this.next = next;
    }
}

public ListNode removingAntipenultimate(ListNode head, int n) {
    ListNode temp = new ListNode(0);
    temp.next = head;
    ListNode left = temp;
    ListNode right = temp;

    for (int i = 0; i <= n; i++) {
    if (right.next != null) {
    right = right.next;
    }
    }
    // Looking for the penultimate n+1 Nodes
    while (right != null) {
    left = left.next;
    right = right.next;
    }
    // Delete the last n Nodes
    left.next = left.next.next;
    return temp.next;
}


static void print(ListNode listNode) {

    while (listNode != null) {
    if (listNode.next == null) {
    System.out.print(listNode.val);
    } else {
    System.out.print(listNode.val + "->");
    }
    listNode = listNode.next;
    }
}
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the Number of Elements");
    int n = sc.nextInt();
    ListNode node = new ListNode(4);
```

```
        ListNode nextnode = node;

        for (int i = 1; i < n; i++) {
        int t = sc.nextInt();
        ListNode nodex = new ListNode(t);

        nextnode.next = nodex;
        nextnode = nextnode.next;
        }
         print(node);
        System.out.println();
        LinkedList LinkedList = new LinkedList();
        ListNode res = LinkedList.removingAntipenultimate(node, 3);
        print(res);
    }
}
```

Output: