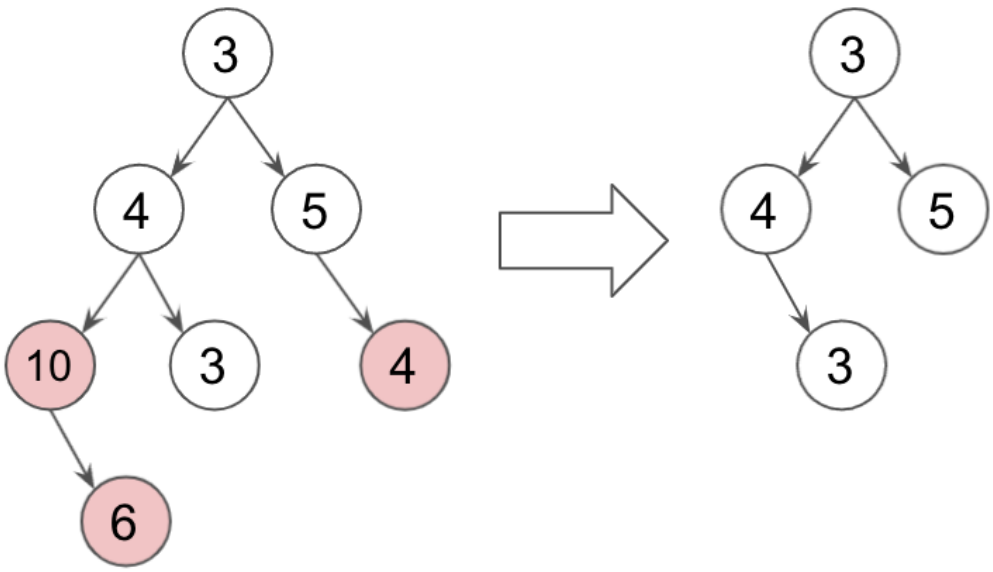
Binary Tree Deletion

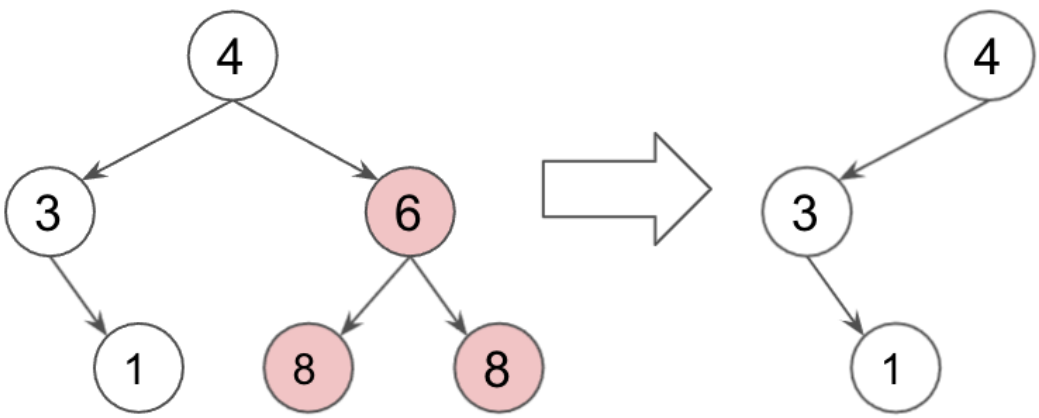
Given a root of a binary tree, delete all subtrees whose elements are all even numbers, and return the modified tree. A subtree of a node is the node itself plus all its children.

The following picture is an illustration of the three samples shown below.

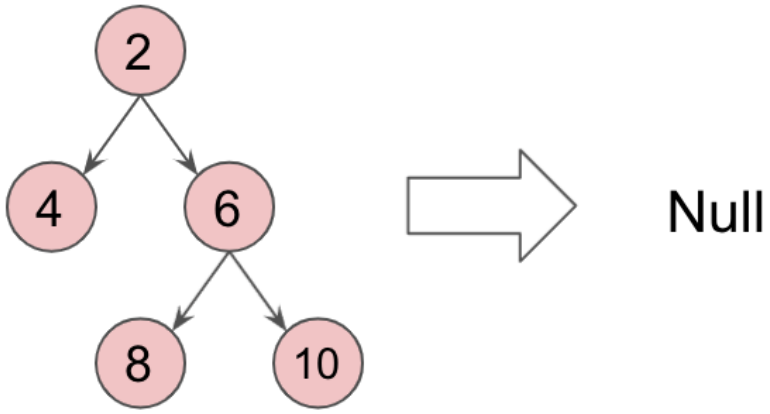
Sample One:



Sample Two:



Sample Three:



Input Format

The input includes two lines. The first line has a single integer: N The second line has N integers or nulls: A[0] A[1] ... A[N-1] The array A stores the input binary tree values by level.

Constraints

1 ≤ N ≤ 500 For each i, 0 ≤ A[i] < 100

Output Format

Output a single line with integers or nulls separated by spaces. The output integers or nulls show the values of the output binary tree by level.

Sample Input 0

```
9
3 4 5 10 3 null 4 null 6
```

Sample Output 0

```
3 4 5 null 3
```

Sample Input 1

```
7
4 3 6 null 1 8 8
```

Sample Output 1

```
4 3 null null 1
```

Sample Input 2

```
7
2 4 6 null null 8 10
```

Sample Output 2

```
null
```

Contest ends in 2 months

Submissions: 24
Max Score: 8

Difficulty: Medium

Rate This Challenge:

More

Java 8

```java
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {
    static class TreeNode {
        int val;
        TreeNode left;
        TreeNode right;
        TreeNode() {}
        TreeNode(int val) { this.val = val; }
        TreeNode(int val, TreeNode left, TreeNode right) {
            this.val = val;
            this.left = left;
            this.right = right;
        }
     }


    // FILL ME IN!
    public static TreeNode deleteTree(TreeNode root) {
        return null;
    }

    public static void main(String[] args) throws IOException {
        // Read input array A. We avoid java.util.Scanner, for speed.
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int N = Integer.parseInt(br.readLine()); // first line
        Integer[] A = new Integer[N];
        StringTokenizer st = new StringTokenizer(br.readLine()); // second line
        for (int i=0; i<N; ++i) {
            String s = st.nextToken();
            A[i] = (s.equals("null") ? null : Integer.parseInt(s));
        }

        // Create the input binary tree
        TreeNode root = new TreeNode();
        if (A[0] == null) {
            root = null;
        }
        else {
            int count = 0;
            Queue<TreeNode> q = new LinkedList<TreeNode>();
            root = new TreeNode(A[0]);
            q.add(root);
            TreeNode cur = null;
            for(int i = 1; i < A.length; i++){
                TreeNode node = new TreeNode();
                if (A[i] == null) {
                    node = null;
                } else {
                    node = new TreeNode(A[i]);
                }
                if(count == 0){
                   cur = q.poll();
                }
                if(count==0){
                  count++;
                  cur.left = node;
                }else {
                  count = 0;
```

```
                    cur.right = node;
                }
                if(A[i] != null){
                    q.add(node);
                }
            }
        }

        // Solve the problem!
        root = deleteTree(root);

        // Print the output binary tree, again buffered for speed.
        PrintWriter out = new PrintWriter(System.out);

        Queue<TreeNode> curr=new LinkedList<TreeNode>();
        Queue<TreeNode> next=new LinkedList<TreeNode>();

        if (root == null) out.print("null ");
        else {
            curr.add(root);
            next.add(root.left);
            next.add(root.right);
            out.print(root.val + " ");
            boolean end = false;
            while (!next.isEmpty()) {
                curr = next;
                next = new LinkedList<TreeNode>();
                while (!curr.isEmpty()) {
                    TreeNode temp = curr.poll();
                    if (temp == null) {
                        end = true;
                        for (TreeNode t : curr) {
                            if (t != null) {
                                end = false;
                                break;
                            }
                        }
                        if (end == true) {
                            for (TreeNode t : next) {
                                if (t != null) {
                                    end = false;
                                    break;
                                }
                            }
                        }
                        if (end == true) break;
                        out.print("null ");
                    } else {
                        out.print(temp.val + " ");
                        next.add(temp.left);
                        next.add(temp.right);
                    }
                }
                if (end == true) break;
            }
        }
        out.close();
    }
}
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code