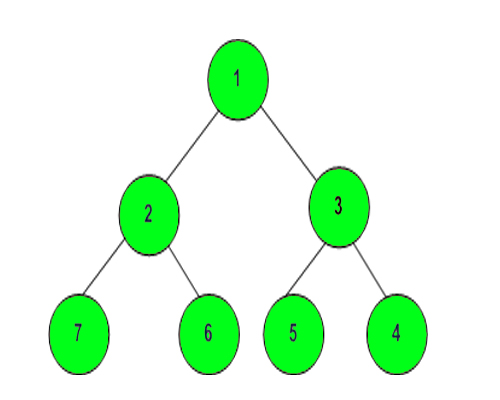
### Level order traversal in spiral form

Given a binary tree and the task is to find the spiral order traversal of the tree.

**Spiral order Traversal mean:** Starting from level 0 for root node, for all the even levels we print the node's value from right to left and for all the odd levels we print the node's value from left to right.

For below tree, function should return 1, 2, 3, 4, 5, 6, 7.  
  
  
****

**Example 1:**

**Input:**

      1

   /   \

  3     2

**Output:**1 3 2

**Example 2:**

**Input:**

           10

        /     \

       20     30

     /    \

   40     60

**Output:** 10 20 30 60 40

**Company Tags**

[**Flipkart**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Flipkart) [**Morgan Stanley**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Morgan%20Stanley) [**Accolite**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Accolite) [**Amazon**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Amazon)[**Microsoft**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Microsoft) [**Hike**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Hike) [**Housing.com**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Housing.com) [**MakeMy Trip**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=MakeMyTrip) [**Ola Cabs**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Ola%20Cabs) [**Payu**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Payu) [**Teradata**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Teradata) [**Walmart**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Walmart) [**Adobe**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Adobe)

**Topic Tags**

[**Recursion**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Recursion) [**Stack**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Stack)[**Tree**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Tree) [**Data Structures**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Data%20Structures) [**Algorithms**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Algorithms)

import java.util.LinkedList;

import java.util.Queue;

import java.io.\*;

import java.util.\*;

class Node{

int data;

Node left;

Node right;

Node(int data){

this.data = data;

left=null;

right=null;

}

}

class CodingMAxima {

static Node buildTree(String str){

if(str.length()==0 || str.charAt(0)=='N'){

return null;

}

String ip[] = str.split(" ");

// Create the root of the tree

Node root = new Node(Integer.parseInt(ip[0]));

// Push the root to the queue

Queue<Node> queue = new LinkedList<>();

queue.add(root);

// Starting from the second element

int i = 1;

while(queue.size()>0 && i < ip.length) {

// Get and remove the front of the queue

Node currNode = queue.peek();

queue.remove();

// Get the current node's value from the string

String currVal = ip[i];

// If the left child is not null

if(!currVal.equals("N")) {

// Create the left child for the current node

currNode.left = new Node(Integer.parseInt(currVal));

// Push it to the queue

queue.add(currNode.left);

}

// For the right child

i++;

if(i >= ip.length)

break;

currVal = ip[i];

// If the right child is not null

if(!currVal.equals("N")) {

// Create the right child for the current node

currNode.right = new Node(Integer.parseInt(currVal));

// Push it to the queue

queue.add(currNode.right);

}

i++;

}

return root;

}

void inOrder(Node node) {

if (node == null) {

return;

}

inOrder(node.left);

System.out.print(node.data + " ");

inOrder(node.right);

}

public static void main (String[] args) throws IOException{

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int t=Integer.parseInt(br.readLine());

while(t-- > 0){

String s = br.readLine();

Node root = buildTree(s);

Spiral g = new Spiral();

ArrayList<Integer> result = g.findSpiral(root);

for(int value : result)

System.out.print(value + " ");

System.out.println();

}

}

}

class Spiral

{

//Function to return a list containing the level order

//traversal in spiral form.

ArrayList<Integer> findSpiral(Node root)

{

ArrayList<Integer> ar=new ArrayList<Integer>();

if(root==null)

return ar;

Queue<Node> q=new LinkedList<>();

// Queue<Integer> level=new LinkedList<>();

Stack<Node> st=new Stack<>();

int n=1;

// level.add(1);

q.add(root);

while(!q.isEmpty()){

int size=q.size();

for(int i=0;i<size;i++){

Node temp=q.poll();

if(n%2==0){

ar.add(temp.data);

}

else{

st.add(temp);

}

if(temp.left!=null)

q.add(temp.left);

if(temp.right!=null)

q.add(temp.right);

}

while(!st.isEmpty()){

ar.add(st.pop().data);

}

n++;

}

return ar;

}

}