**ROUND 2:**

**Q.** **Given root of a binary tree, return the right side view of the tree (from top to bottom)**Ex.            1     <-

    2         3   <-

          4  8     5    6 <-

Return [1,3,6]

struct Node {

Int val;

Node \*left;  
 Node \*right;  
}

-------------------------------

Void print\_right(Node \*root, vector<int> &ans, int height)

{

if(root == NULL)

Return;

If (height == ans.size())

ans.push\_back(root->val);

print\_right(root->right, ans, height + 1);

print\_right(root->left, ans, height + 1);

return;

}

vector<int> ans;

print\_right(root, ans, 0);

**Q**. **Given a graph (binary adjacency matrix) find the sinkhole in it. If no sinkhole exists return -1;  
Sinkhole: it has no outgoing edge. Every other node in the graph has an edge to this sinkhole**  
  
Ex. 1->2, 4->2, 1->4, 3->2   
-> return 2

   1 2 3 4

1 0 1 0 1

2 0 0 0 0

3 0 1 0 0

4 0 1 0 0

Int find\_sinkhole(vector<vector<int>> graph)

{

Int  n = graph.size();

Int current = 0;

For (int i = 1; i < n; i++)

{

If (graph[current][i] == 1)

Current = i;

}

For (int i = 0; i < n; i++)

{

If (graph[current][i] == 1)

Return -1;

}

For (int j = 0; j < n; j++)

{

If (current != j && graph[j][current] == 0)

Return -1;

}

Return current;

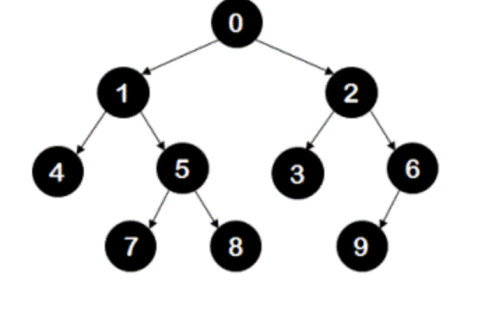
}

**Round 1 questions:**

<https://www.geeksforgeeks.org/zigzag-tree-traversal/>

<https://www.geeksforgeeks.org/find-the-next-lexicographically-greater-word-than-a-given-word/>

<https://www.geeksforgeeks.org/print-nodes-top-view-binary-tree/>



0 2 1 4 5 3 6 9 8 7

Struct Node

{

Int key;

Node\* left;

Node\* right;

};

Void print\_tree(Node \*root)

{

stack<Node\*> even, odd;

even.push(root);

While (!even.empty() || !odd.empty())

{

If (!even.empty())

{

While (!even.empty())

{

Node\* temp = even.top();

even.pop();

Cout << temp->key << “ “;

If (temp->left)

odd.push(temp->left);

If (temp->right)

odd.push(temp->right);

}

}

Else

{

}

}

}

Akljdefghi

Adcb -- bdca ->  bacd

8

1 0

8         3   6                  9

7             5 a

          B

c