Practice Questions

Solution 1:-

```
class Solution {
   // Function to return a list containing the inorder traversal of the
tree.
   ArrayList<Integer> inOrder(Node root) {
      ArrayList<Integer>a=new ArrayList<>();
      sol(a,root);
      return a;
   }
   public void sol(ArrayList<Integer>a,Node root){
      if(root==null){
         return;
      }
      else{
      sol(a,root.left);
      a.add(root.data);
      sol(a,root.right);
      }
   }
}
```

Solution 2:-

```
class Solution
{
   // Return True if the given trees are isomotphic. Else return False.
   boolean isIsomorphic(Node root1, Node root2)
```

```
    {
         // code here.

          if (root1 == null && root2 == null) {
            return true;
        }

        if (root1 == null || root2 == null) {
            return false;
        }
        if (root1.data != root2.data) {
            return false;
        } else {
            return (isIsomorphic(root1.left, root2.left) &&
isIsomorphic(root1.right, root2.right)) || (isIsomorphic(root1.left,
root2.right) && isIsomorphic(root1.right, root2.left));
        }


    }

}
```

Solution 3:-

```
class Solution

{

    private static Node th = null, tt = null;

    private static void addFirstNode(Node head){

        if (th == null) {

            th = head;

            tt = head;

        } else {
```

```
        head.next = th;
        th = head;
    }
}
public static Node reverseBetween(Node head, int n, int m)
{
    if(head == null || head.next == null || n == m){
        return head;
    }
    th = null;
    tt = null;
    Node dummy = new Node(-1), prev = dummy, curr = head;
    prev.next = head;
    int i = 1;
    while(i <= m){
        while(i >= n && i <= m){
            Node forw = curr.next;
            curr.next = null;
            addFirstNode(curr);
            curr = forw;
            i++;
        }
        if(i > m){
            prev.next = th;
            tt.next = curr;
            break;
        }
```

```
          i++;
          prev = curr;
          curr = curr.next;
       }
       return dummy.next;
   }
}
```

Solution 4:-

```
class Solution {
   Node reorderlist(Node head) {
      Node curr=head;
      List<Integer> list=new ArrayList();
      while(curr!=null)
      {
         list.add(curr.data);
         curr=curr.next;
      }
      Node node=new Node(0);
      int j=0;
      int k=list.size()-1;

      int[] val=new int[list.size()];
      for(int i=0;i<list.size();i++)
      {
         if(i%2==0)
         {
            val[i]=list.get(j);
            j++;
         }
         else if(i%2!=0)
         {
                val[i]=list.get(k);
                k--;
            }
        }
       // System.out.println(Arrays.toString(val));
       Node nodes=node;
        for(int vals:val)
        {
            nodes.next=new Node(vals);
            nodes=nodes.next;
        }
        return node.next;
    }
}
```