- Daily Practice Program   Date 1/12/2023

Problem 1: Day of the year

```
package ishwarchavan.com;

public class DayOfTheYear {      //class created

	static int days [] = { 31, 28, 31, 30, 31, 30,
	                              31, 31, 30, 31, 30, 31 };

	static int dayOfYear(String date){     //function created

		int year = Integer.parseInt(date.substring(0, 4));    // Extract the year, 
month and the day from the date string

		int month = Integer.parseInt(date.substring(5, 7));

		int day = Integer.parseInt(date.substring(8));

		if (month > 2 && year % 4 == 0 && (year % 100 != 0 || year % 400 == 0)){
// If current year is a leap year and the dategiven is after the 28th of February then
                                                                               // it 
must include the 29th February
			++day;
		}
		while (--month > 0){     // Add the days in the previous months
			day = day + days[month - 1];
		}
		return day;
	}

	public static void main (String[] args){     //main program created
		String date = "2019-01-09";
		System.out.println(dayOfYear(date));
	}
}
```

Problem 2: Univalued Binary Tree

```
package ishwarchavan.com;
import java.io.*;
import java.util.*;

public class UnivaluedBinaryTree {     //class created

	static class Node {  // Structure of a tree node
		int data;
		Node left;
		Node right;
	};

	static Node newNode(int data)        // Function to insert a new node in a binary 
tree
	{
		Node temp = new Node();
		temp.data = data;
		temp.left = temp.right = null;
		return (temp);
	}

	static boolean isUnivalTree(Node root){    // Function to check If the tree is 
univalued or not

		if (root == null) {  // If tree is an empty tree
			return true;
		}
```

```
            Queue<Node> q = new LinkedList<>();  // Store nodes at each level of the
tree

            q.add(root);  // Insert root node

            int rootVal = root.data;  // Stores value of root node

            while (!q.isEmpty()) {    // Traverse the tree using BFS

                  Node currRoot = q.peek();  // Stores front element of the queue

                  if (currRoot.data != rootVal) {     // If value of traversed node not
equal to value of root node
                        return false;
                  }

                  if (currRoot.left != null) {   // If left subtree is not NULL

                        q.add(currRoot.left);  // Insert left subtree
                  }

                  if (currRoot.right != null) {    // If right subtree is not NULL

                        q.add(currRoot.right);        // Insert right subtree
                  }
                  q.remove();   // Remove front element of the queue
            }

            return true;
      }
      public static void main(String[] args){    //main program created

            Node root = newNode(1);
            root.left = newNode(1);
            root.right = newNode(1);
            root.left.left = newNode(1);
            root.left.right = newNode(1);
            root.right.right = newNode(1);

            if (isUnivalTree(root)) {        //condition checking
                  System.out.print("YES");
            }
            else {
                  System.out.print("NO");
            }
      }
}
```