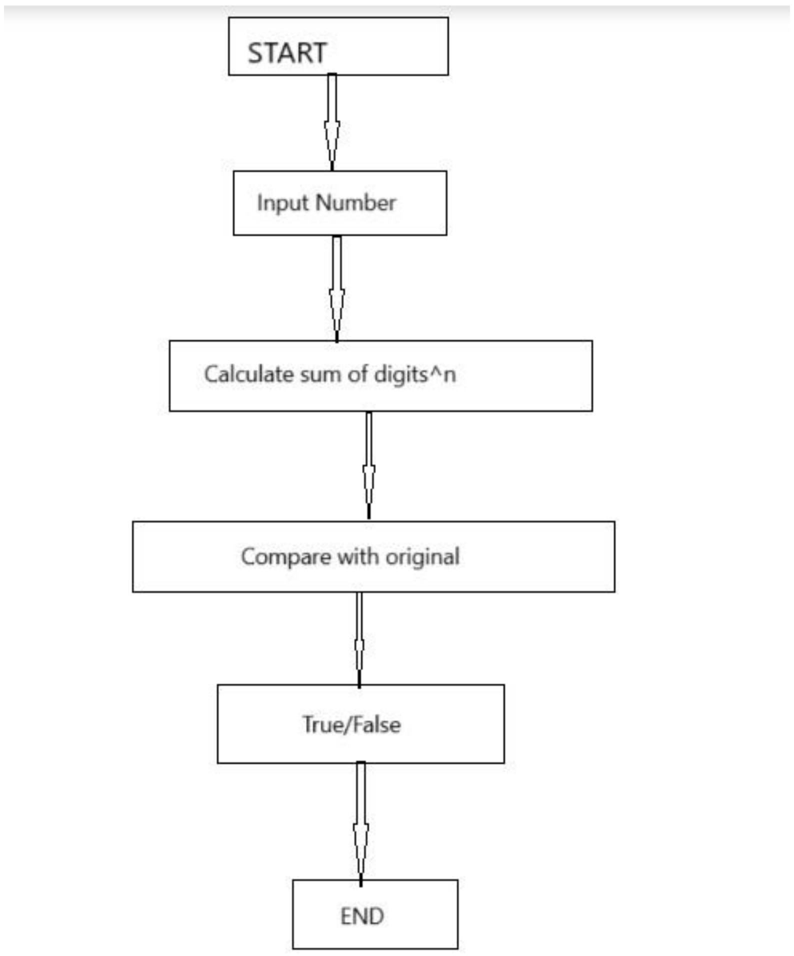
C-DAC Mumbai

Date 25/09/2024

Subject: Algorithm and Data Structure
Assignment 1

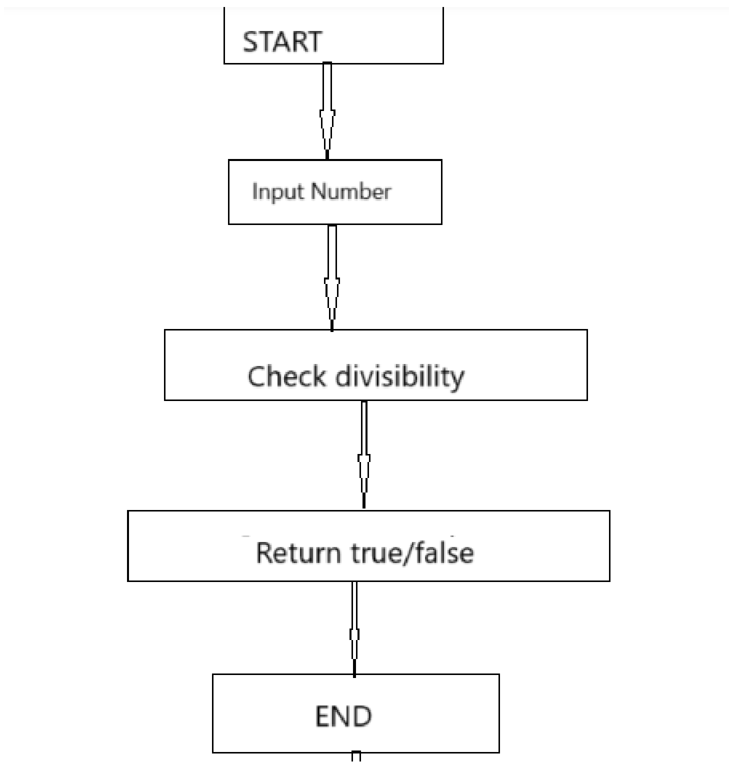
Q.1



```
public class ArmstrongNumber {

    public static boolean isArmstrong(int number) {

        int original = number;

        int sum = 0;
        int digits = String.valueOf(number).length();


        while (number != 0) {

            int digit = number % 10;

            sum += Math.pow(digit, digits);

            number /= 10;

        }

        return sum == original;

    }


    public static void main(String[] args) {

        System.out.println(isArmstrong(153));  // Output: true

        System.out.println(isArmstrong(123));  // Output: false

    }

}
```

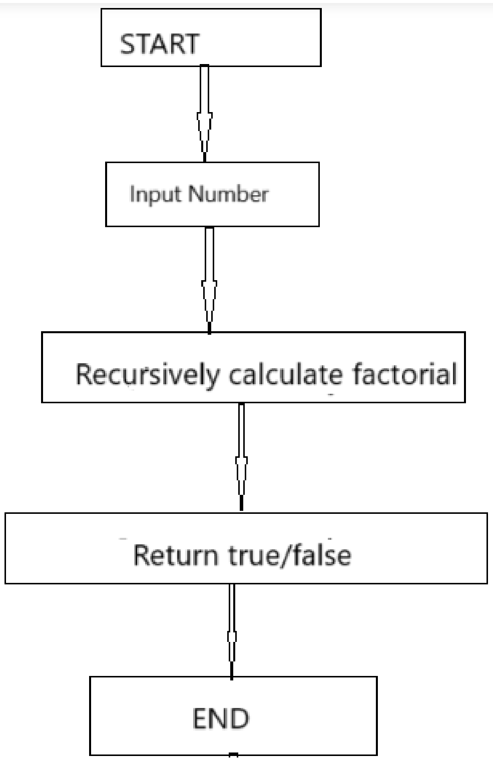
Q.2

}



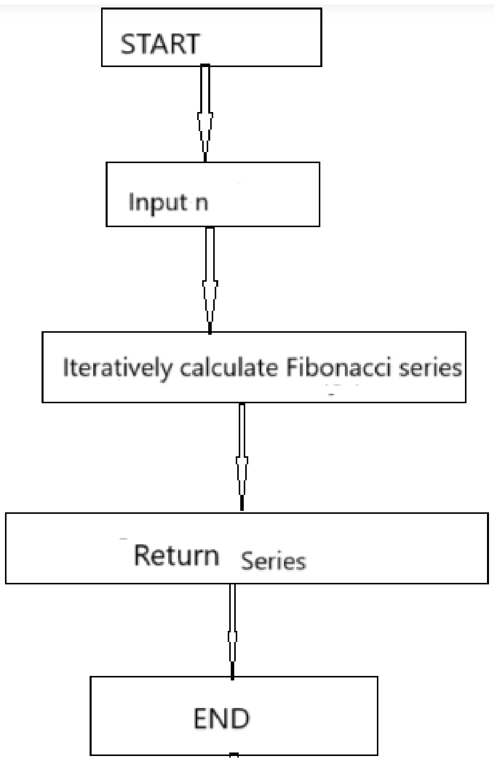
```
public class PrimeNumber {
  public static boolean isPrime(int number) {
    if (number <= 1) return false;
    for (int i = 2; i <= Math.sqrt(number); i++) {
      if (number % i == 0) return false;
    }
    return true;
  }

  public static void main(String[] args) {
    System.out.println(isPrime(29));  // Output: true
    System.out.println(isPrime(15));  // Output: false
  }
}
```

Q.3



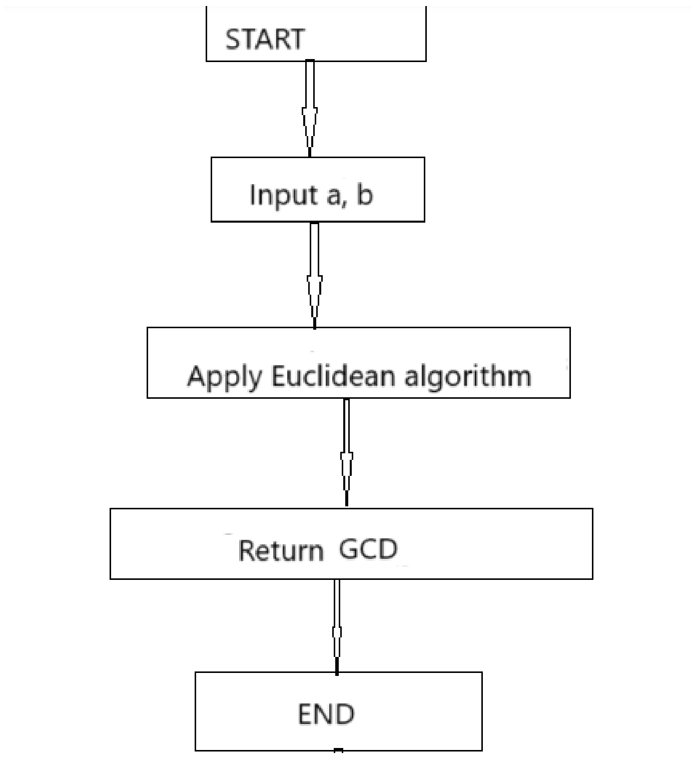
```
public class FactorialRecursive {

   public static int factorial(int n) {

      if (n == 0) return 1;

      return n * factorial(n - 1);

   }


   public static void main(String[] args) {

      System.out.println(factorial(5));  // Output: 120

      System.out.println(factorial(0));  // Output: 1

   }

}
```

Q.4



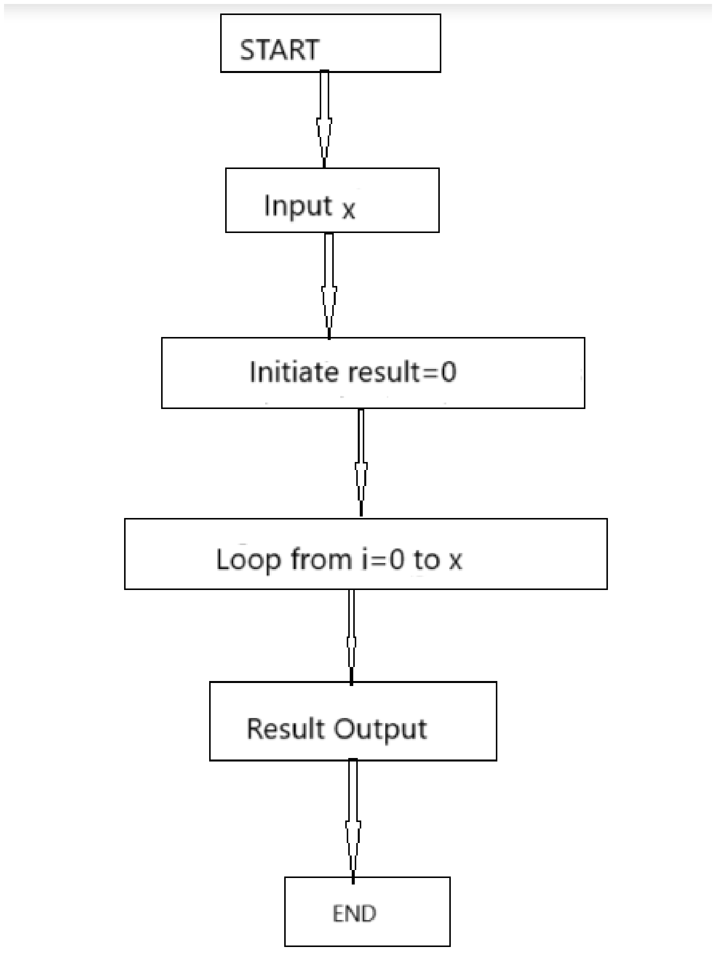
```
import java.util.ArrayList;

import java.util.List;


public class FibonacciRecursive {

  public static int fibonacci(int n) {

    if (n == 0) return 0;

    if (n == 1) return 1;

    return fibonacci(n - 1) + fibonacci(n - 2);

  }


  public static List<Integer> getFibonacciSeries(int n) {

    List<Integer> series = new ArrayList<>();

    for (int i = 0; i < n; i++) {
            series.add(fibonacci(i));
        }
        return series;
    }

    public static void main(String[] args) {
        System.out.println(getFibonacciSeries(5));  // Output: [0, 1, 1, 2, 3]
        System.out.println(getFibonacciSeries(8));  // Output: [0, 1, 1, 2, 3, 5, 8, 13]
    }
}
```

Q.5



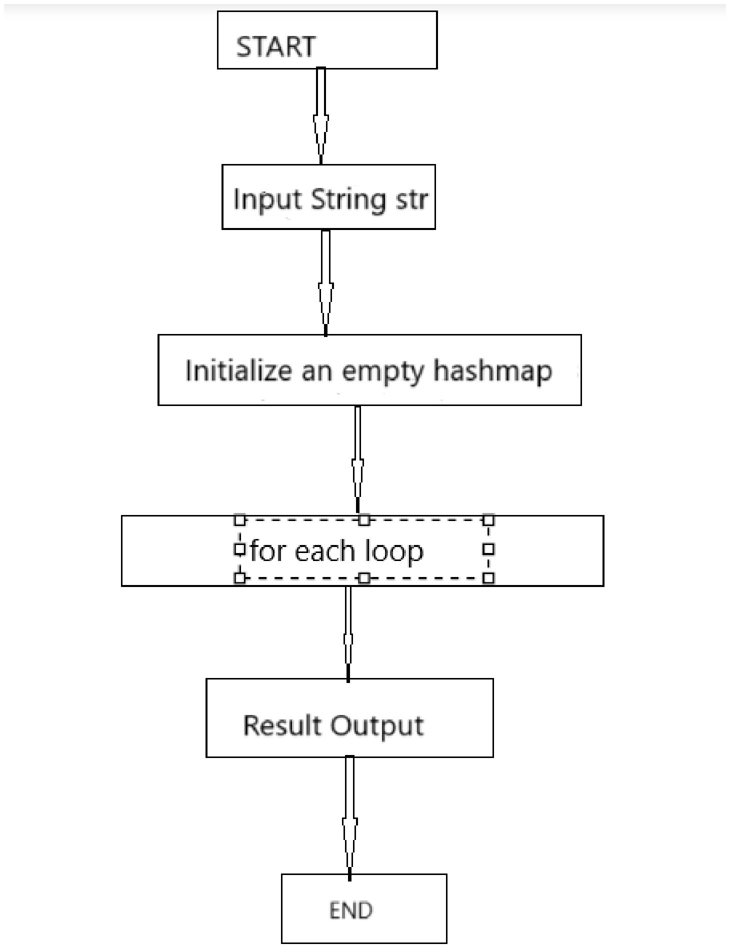
```
public class GCDRecursive {
    public static int gcd(int a, int b) {
        if (b == 0) return a;
        return gcd(b, a % b);
    }

    public static void main(String[] args) {
        System.out.println(gcd(54, 24));  // Output: 6
        System.out.println(gcd(17, 13));  // Output: 1
    }
}
```

Q.6



```
public class SquareRoot {
    public static int sqrt(int x) {
        if (x == 0 || x == 1) return x;
        int start = 1, end = x, result = 0;

        while (start <= end) {
            int mid = (start + end) / 2;

            if (mid * mid == x) return mid;

            if (mid * mid < x) {
                start = mid + 1;

                result = mid;

            } else {

                end = mid - 1;

            }

        }

        return result;

    }


    public static void main(String[] args) {

        System.out.println("Square root of 16: " + sqrt(16)); // Output: 4

        System.out.println("Square root of 27: " + sqrt(27)); // Output: 5

    }

}
```

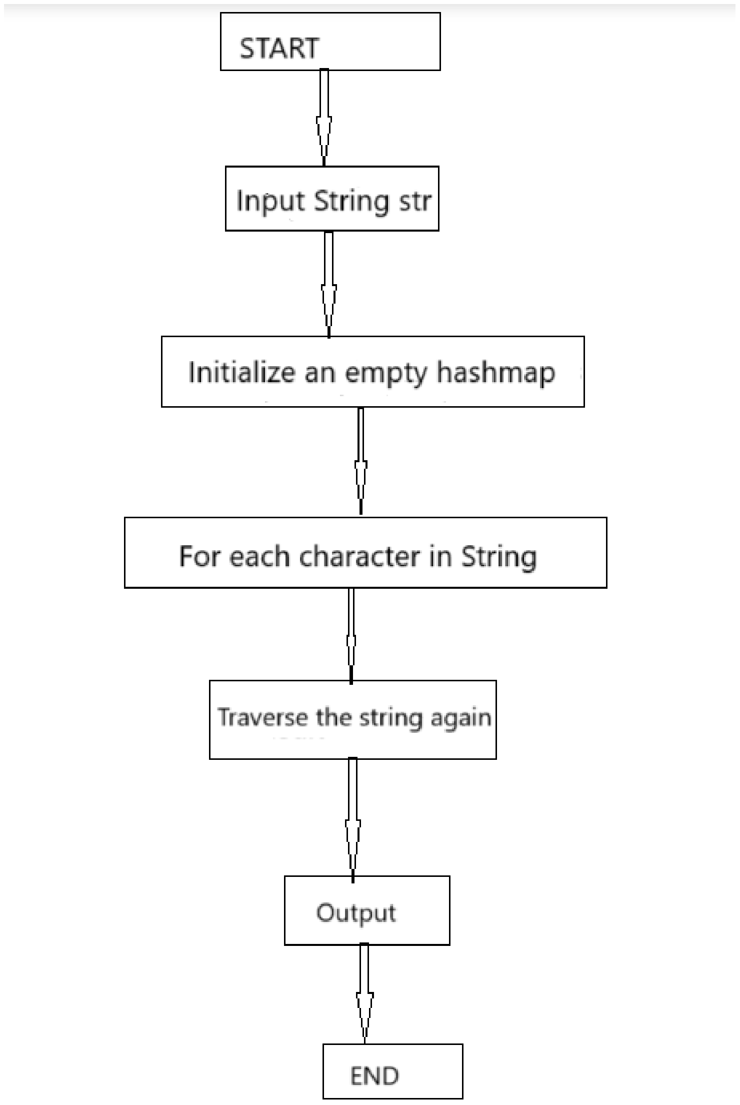
Q.7



```
import java.util.*;


public class RepeatedCharacters {
   public static List<Character> findRepeatedChars(String str) {
      Map<Character, Integer> charCount = new HashMap<>();
      List<Character> repeatedChars = new ArrayList<>();


      for (char c : str.toCharArray()) {
         charCount.put(c, charCount.getOrDefault(c, 0) + 1);
      }
```

```
        for (Map.Entry<Character, Integer> entry : charCount.entrySet()) {
            if (entry.getValue() > 1) {
                repeatedChars.add(entry.getKey());
            }
        }
        return repeatedChars;
    }

    public static void main(String[] args) {
        System.out.println(findRepeatedChars("programming")); // Output: [r, g, m]
        System.out.println(findRepeatedChars("hello"));       // Output: [l]
    }
}
```

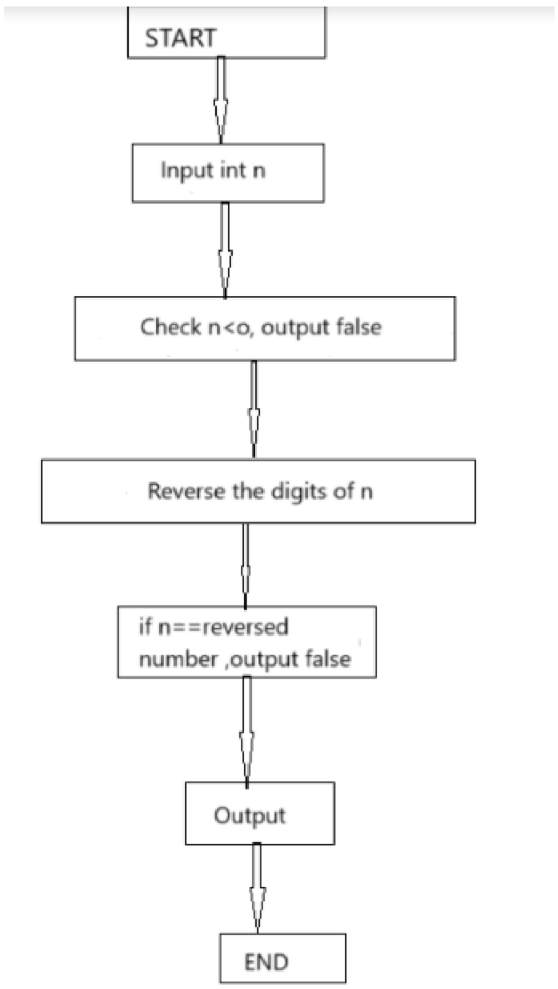
Q.8


START
Input String str
Initialize an empty hashmap
For each character in String
Traverse the string again
Output
END

```
import java.util.*;


public class FirstNonRepeated {

   public static Character findFirstNonRepeatedChar(String str) {

      Map<Character, Integer> charCount = new LinkedHashMap<>();


      for (char c : str.toCharArray()) {

         charCount.put(c, charCount.getOrDefault(c, 0) + 1);

      }
```

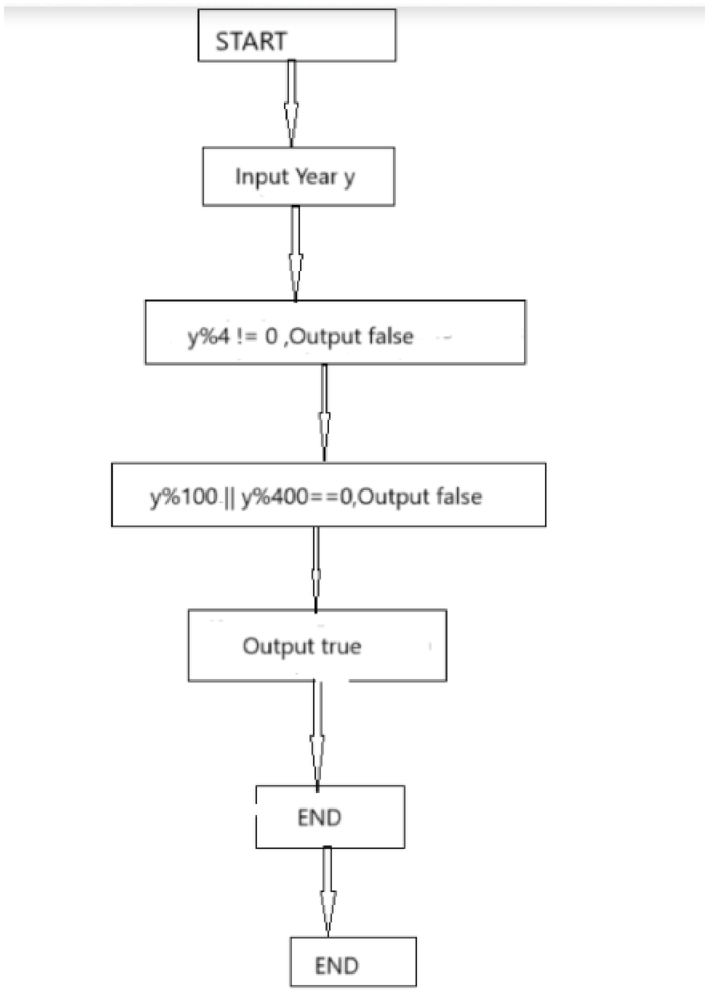
```
        for (Map.Entry<Character, Integer> entry : charCount.entrySet()) {
            if (entry.getValue() == 1) {
                return entry.getKey();
            }
        }

        return null;
    }

    public static void main(String[] args) {
        System.out.println(findFirstNonRepeatedChar("stress"));  // Output: 't'
        System.out.println(findFirstNonRepeatedChar("aabbcc"));  // Output: null
    }
}
```

Q.9



```
public class IntegerPalindromeRecursion {

  public static boolean isPalindromeHelper(int n, int temp) {

    if (n == 0) return temp == 0;

    temp = temp * 10 + n % 10;

    return isPalindromeHelper(n / 10, temp);

  }


  public static boolean isPalindrome(int n) {

    if (n < 0) return false;  // Negative numbers cannot be palindromes

    return isPalindromeHelper(n, 0);
    }

    public static void main(String[] args) {
        System.out.println(isPalindrome(121)); // Output: true
        System.out.println(isPalindrome(-121)); // Output: false
    }
}
```

Q.10



```
public class LeapYear {
    public static boolean isLeapYear(int year) {
        if (year % 4 == 0) {
            if (year % 100 == 0) {
                return year % 400 == 0;
            } else {
                return true;
            }
        }
        return false;
    }

    public static void main(String[] args) {
        System.out.println(isLeapYear(2020));  // Output: true
        System.out.println(isLeapYear(1900));  // Output: false
    }
}
```