Subject: Algorithm and Data Structure

Assignment 1

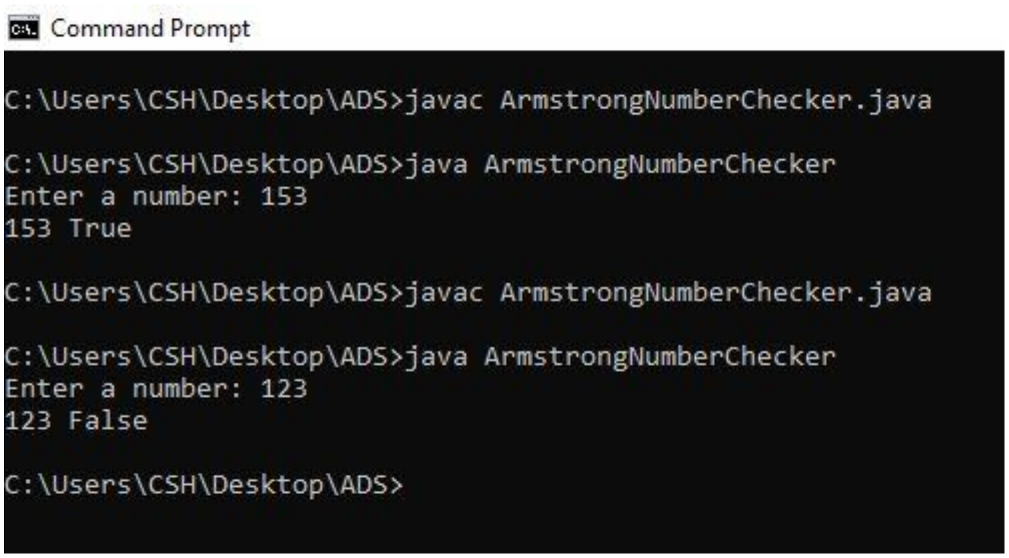
1. Armstrong Number
Problem: Write a Java program to check if a given number is an Armstrong number.

Test Cases:

Input: 153
Output: true
Input: 123
Output: false

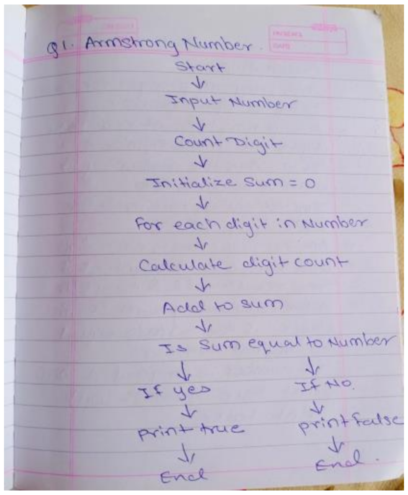
Code –
```
class ArmstrongNumberChecker {

    public static boolean isArmstrong(int number) {
        int originalNumber = number;
        int result = 0;
        int numberOfDigits = String.valueOf(number).length();

        while (number != 0) {
            int digit = number % 10;
            result += Math.pow(digit, numberOfDigits);
            number /= 10;
        }

        return result == originalNumber;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a number: ");
        int number = scanner.nextInt();

        if (isArmstrong(number)) {
            System.out.println(number + " True");
        } else {
            System.out.println(number + " False");
        }

        scanner.close();
    }
}
```

Output –

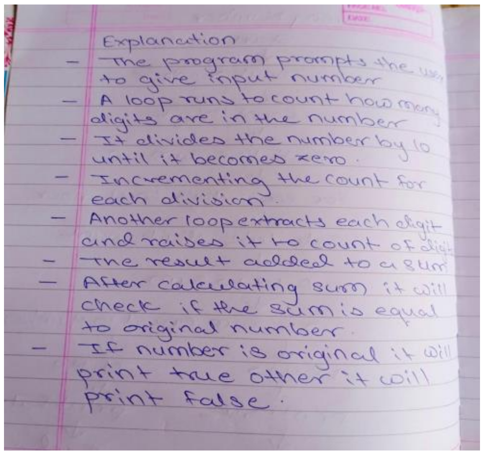


```
C:\Users\CSH\Desktop\ADS>javac ArmstrongNumberChecker.java

C:\Users\CSH\Desktop\ADS>java ArmstrongNumberChecker
Enter a number: 153
153 True

C:\Users\CSH\Desktop\ADS>javac ArmstrongNumberChecker.java

C:\Users\CSH\Desktop\ADS>java ArmstrongNumberChecker
Enter a number: 123
123 False

C:\Users\CSH\Desktop\ADS>
```

Flowchar –



Explanation-



Explanation
- The program prompts the user to give input number
- A loop runs to count how many digits are in the number
- It divides the number by 10 until it becomes zero.
- Incrementing the count for each division.
- Another loop extracts each digit and raises it to count of digit
- The result added to a sum
- After calculating sum it will check if the sum is equal to original number.
- If number is original it will print true other it will print false.

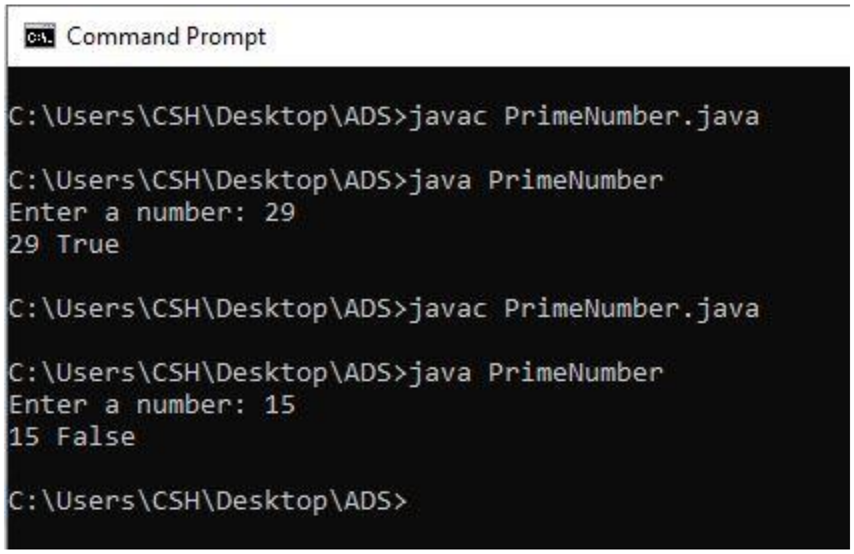
2. Prime Number
Problem: Write a Java program to check if a given number is prime.

Test Cases:
Input: 29
Output: true
Input: 15
Output: false

Code –

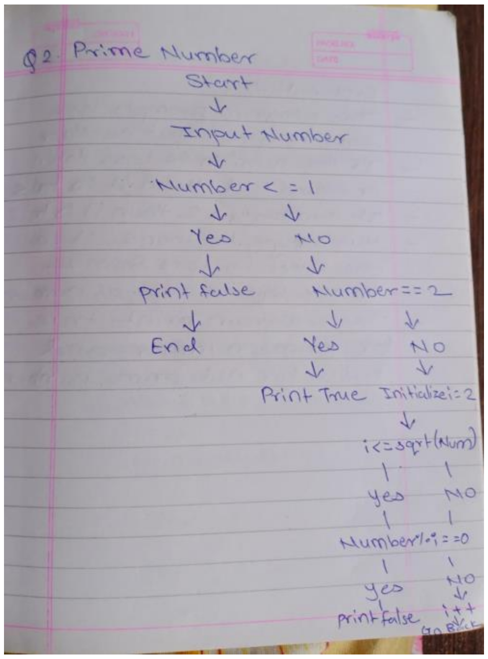
```
import java.util.Scanner;



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) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a number: ");
        int number = scanner.nextInt();

        if (isPrime(number)) {
            System.out.println(number + " True");
        } else {
            System.out.println(number + " False");
        }

        scanner.close();
    }
}
```

Output –

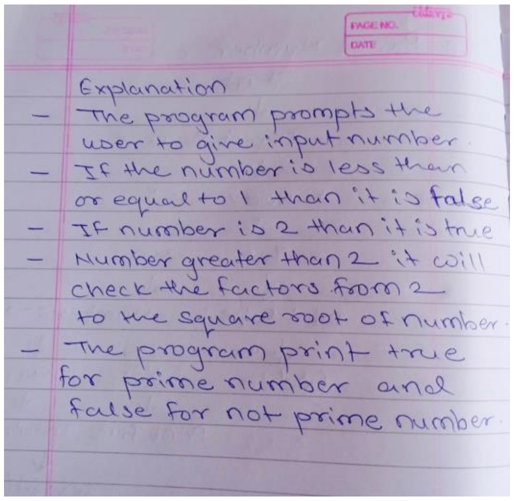


```
C:\Users\CSH\Desktop\ADS>javac PrimeNumber.java

C:\Users\CSH\Desktop\ADS>java PrimeNumber
Enter a number: 29
29 True

C:\Users\CSH\Desktop\ADS>javac PrimeNumber.java

C:\Users\CSH\Desktop\ADS>java PrimeNumber
Enter a number: 15
15 False

C:\Users\CSH\Desktop\ADS>
```

Flowchart –



Explanation -


PAGE NO.
DATE

Explanation
- The program prompts the user to give input number.
- If the number is less than or equal to 1 than it is false
- If number is 2 than it is true
- Number greater than 2 it will check the factors from 2 to the square root of number.
- The program print true for prime number and false for not prime number.

3. Factorial
Problem: Write a Java program to compute the factorial of a given number.

Test Cases:

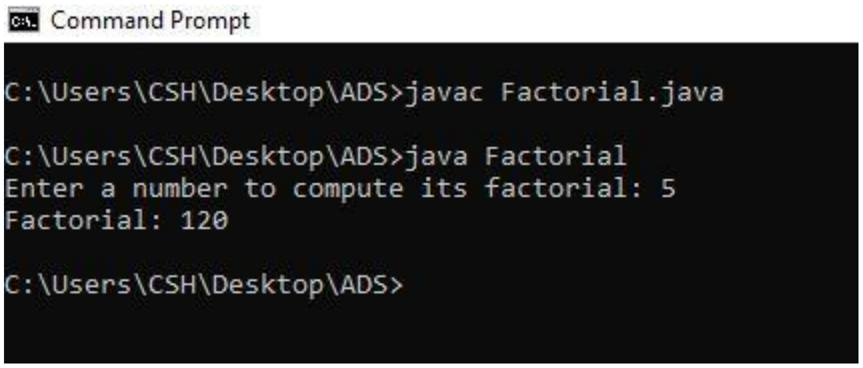
Input: 5
Output: 120
Input: 0
Output: 1

Code –

```
public class Factorial {



  public static long factorial(int number) {

    if (number < 0) {

      throw new IllegalArgumentException("Factorial is not defined for negative numbers.");

    }



    long result = 1;
```

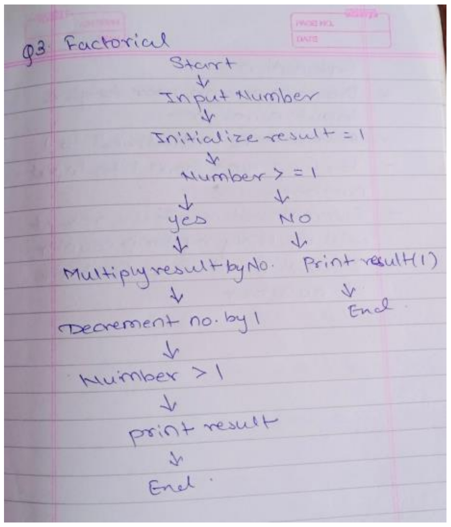
```
        for (int i = 2; i <= number; i++) {
            result *= i;
        }

        return result;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a number to compute its factorial: ");
        int number = scanner.nextInt();

        try {
            System.out.println("Factorial: " + factorial(number));
        } catch (IllegalArgumentException e) {
            System.out.println(e.getMessage());
        }

        scanner.close();
    }
}
```

Output –

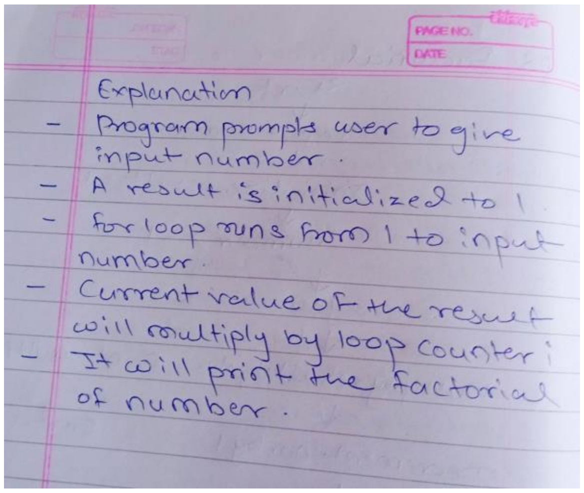


```
C:\Users\CSH\Desktop\ADS>javac Factorial.java

C:\Users\CSH\Desktop\ADS>java Factorial
Enter a number to compute its factorial: 5
Factorial: 120

C:\Users\CSH\Desktop\ADS>
```

Flowchart –



Explanation –



Explanation
- Program prompts user to give input number.
- A result is initialized to 1.
- for loop runs from 1 to input number.
- Current value of the result will multiply by loop counter.
- It will print the factorial of number.

4. Fibonacci Series
Problem: Write a Java program to print the first n numbers in the Fibonacci series.

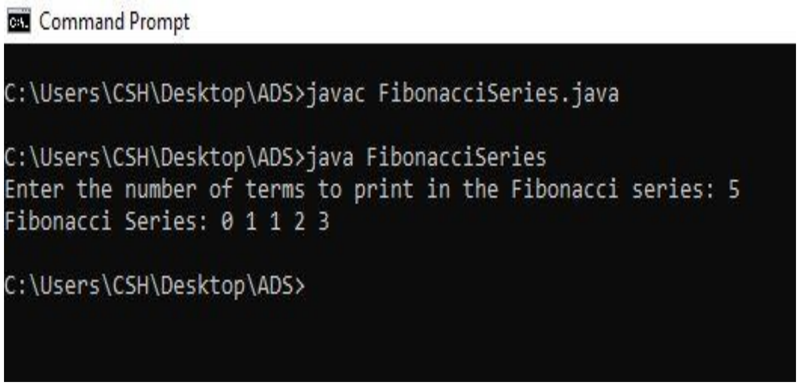
Test Cases:

Input: n = 5
Output: [0, 1, 1, 2, 3]
Input: n = 8
Output: [0, 1, 1, 2, 3, 5, 8, 13]

Code –

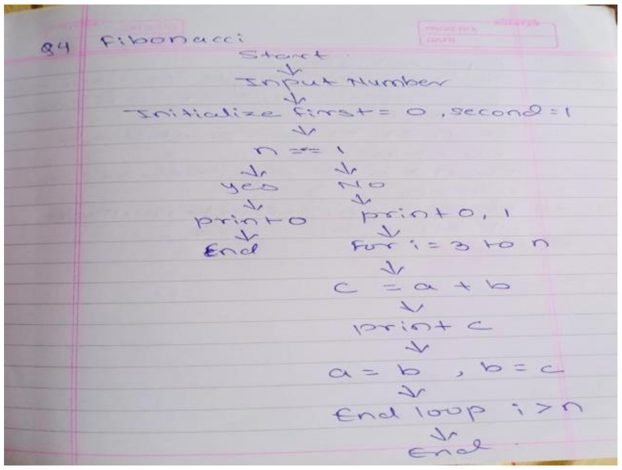
```
 import java.util.Scanner;



public class FibonacciSeries {



    public static void printFibonacci(int n) {

        if (n <= 0) {

            System.out.println("The number of terms must be positive.");

            return;

        }



        int first = 0, second = 1;



        System.out.print("Fibonacci Series: ");

        for (int i = 1; i <= n; i++) {

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



            int next = first + second;

            first = second;
            second = next;
        }
        System.out.println();
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of terms to print in the Fibonacci series: ");
        int n = scanner.nextInt();

        printFibonacci(n);

        scanner.close();
    }
}
```

Output-

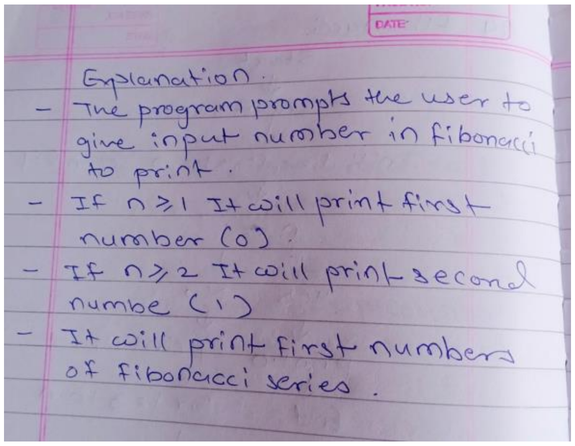


```
C:\Users\CSH\Desktop\ADS>javac FibonacciSeries.java

C:\Users\CSH\Desktop\ADS>java FibonacciSeries
Enter the number of terms to print in the Fibonacci series: 5
Fibonacci Series: 0 1 1 2 3

C:\Users\CSH\Desktop\ADS>
```

Flowchart –



Explanation –



Explanation.

- The program prompts the user to give input number in fibonacci to print.
- If n ≥ 1 It will print first number (0)
- If n ≥ 2 It will print second numbe (1)
- It will print first numbers of fibonacci series.

5. Find GCD

Problem: Write a Java program to find the Greatest Common Divisor (GCD) of two numbers.

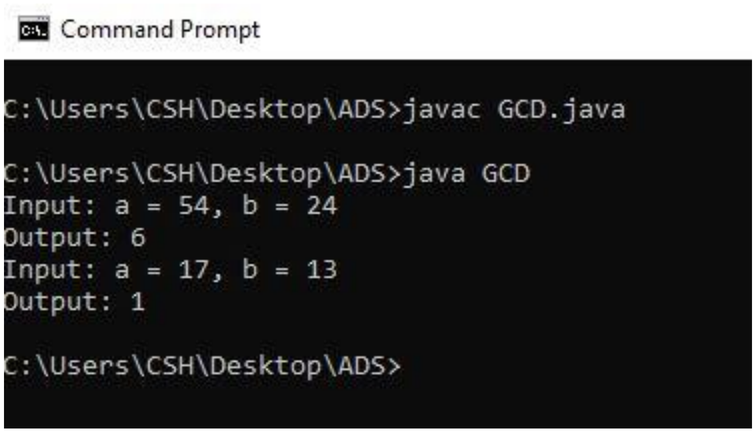
Test Cases:

Input: a = 54, b = 24
Output: 6
Input: a = 17, b = 13
Output: 1

Code –

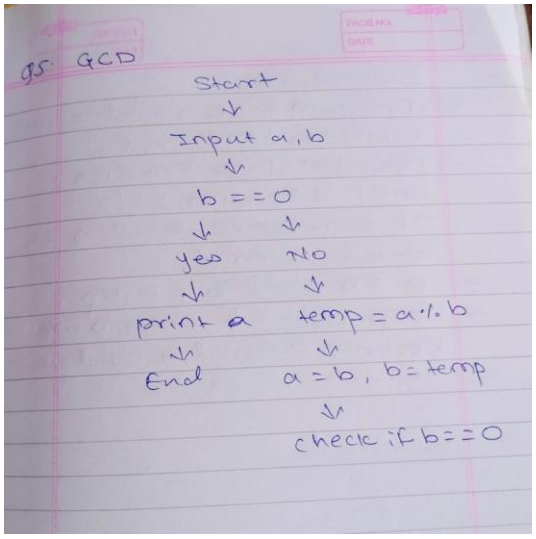
```
public class GCD {



    public static int findGCD(int a, int b) {
        while (b != 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        return a;
    }

    public static void main(String[] args) {

        int a = 54, b = 24;
        System.out.println("Input: a = " + a + ", b = " + b);
        System.out.println("Output: " + findGCD(a, b));

        a = 17;
        b = 13;
        System.out.println("Input: a = " + a + ", b = " + b);
        System.out.println("Output: " + findGCD(a, b));



    }
}
```

Output –

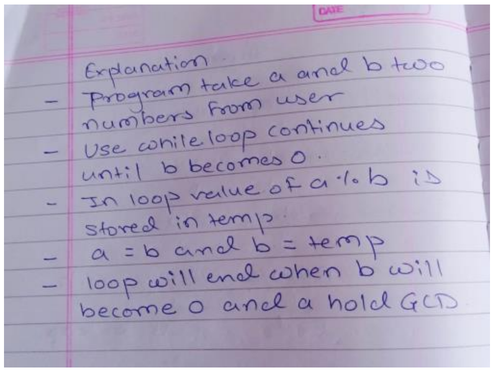


```
C:\Users\CSH\Desktop\ADS>javac GCD.java

C:\Users\CSH\Desktop\ADS>java GCD
Input: a = 54, b = 24
Output: 6
Input: a = 17, b = 13
Output: 1

C:\Users\CSH\Desktop\ADS>
```

Flowchart –



Explanation-



Explanation
- Program take a and b two
  numbers from user
- Use while loop continues
  until b becomes 0.
- In loop value of a % b is
  stored in temp.
- a = b and b = temp
- loop will end when b will
  become 0 and a hold GCD

6. Find Square Root

Problem: Write a Java program to find the square root of a given number (using integer approximation).

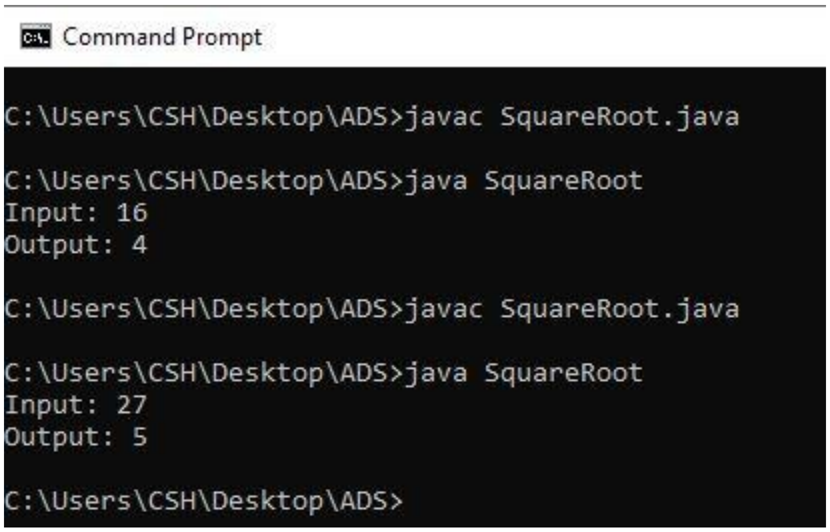
Test Cases:

Input: x = 16
Output: 4
Input: x = 27
Output: 5

Code –

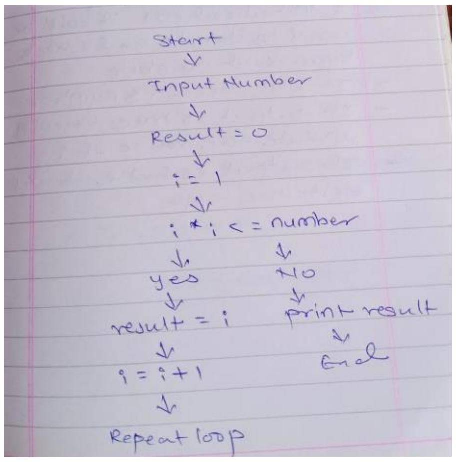
```
import java.util.Scanner;


public class SquareRoot {


    public static int findSquareRoot(int number) {

        int result = 0;


        for (int i = 1; i * i <= number; i++) {

            result = i;

        }


        return result;

    }


    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);


        System.out.print("Input: ");

        int number = scanner.nextInt();


        int squareRoot = findSquareRoot(number);
        System.out.println("Output: " + squareRoot);


        scanner.close();
    }
}
```

Output –

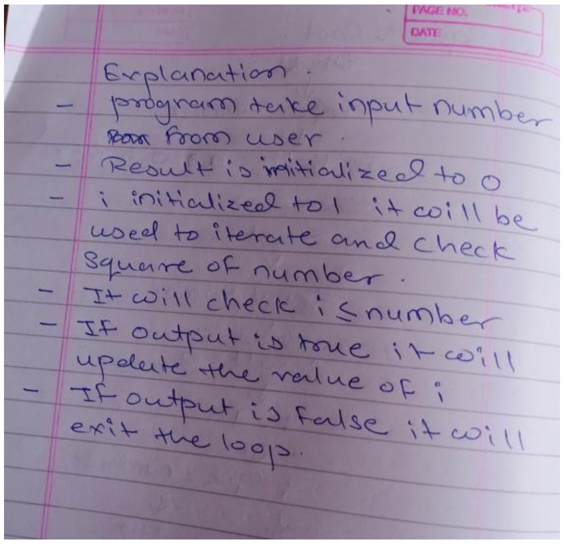


```
C:\Users\CSH\Desktop\ADS>javac SquareRoot.java

C:\Users\CSH\Desktop\ADS>java SquareRoot
Input: 16
Output: 4

C:\Users\CSH\Desktop\ADS>javac SquareRoot.java

C:\Users\CSH\Desktop\ADS>java SquareRoot
Input: 27
Output: 5

C:\Users\CSH\Desktop\ADS>
```

Flowchart –



Explanation-



PAGE NO.
DATE

Explanation.
- program take input number ~~from~~ from user.
- Result is initialized to 0
- i initialized to 1 it will be used to iterate and check square of number.
- It will check i ≤ number
- If output is true it will update the value of i
- If output is false it will exit the loop.

7. Find Repeated Characters in a String
Problem: Write a Java program to find all repeated characters in a string.

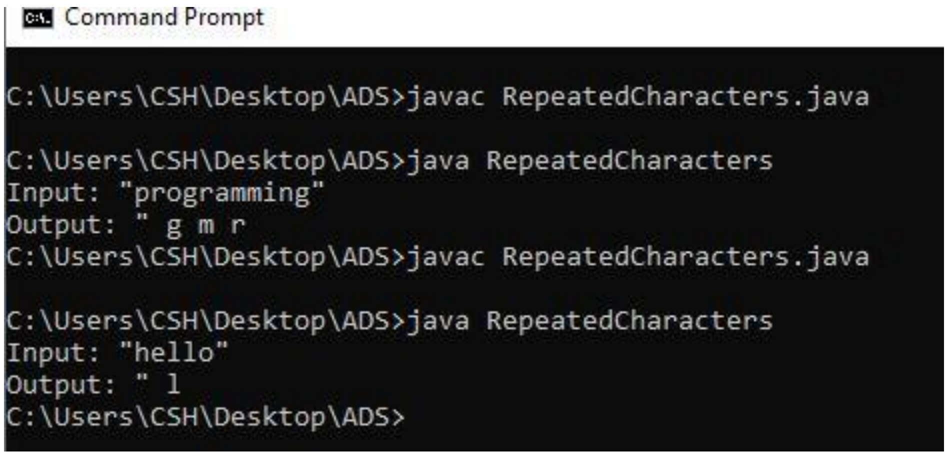
Test Cases:

Input: "programming"
Output: ['r', 'g', 'm']
Input: "hello"
Output: ['l']

Code –

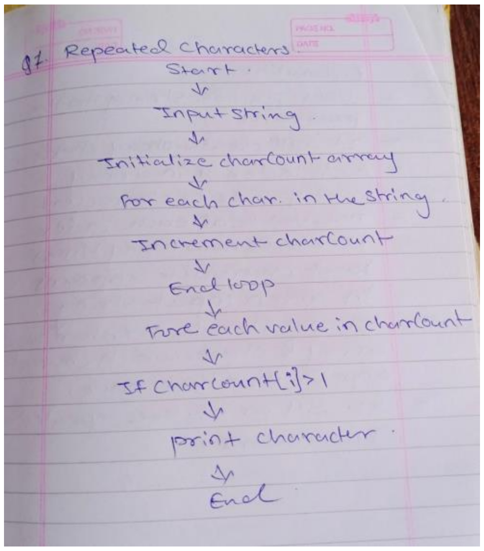
```
import java.util.Scanner;



public class RepeatedCharacters {



    public static void findRepeatedCharacters(String input) {



        int[] charCount = new int[256];
        for (int i = 0; i < input.length(); i++) {
            charCount[input.charAt(i)]++;
        }

        System.out.print("Output: ");
        for (int i = 0; i < 256; i++) {
            if (charCount[i] > 1) {
                System.out.print((char)i + " ");
            }
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Input: ");
        String input = scanner.nextLine();

        findRepeatedCharacters(input);

        scanner.close();
    }
}
```

Output-

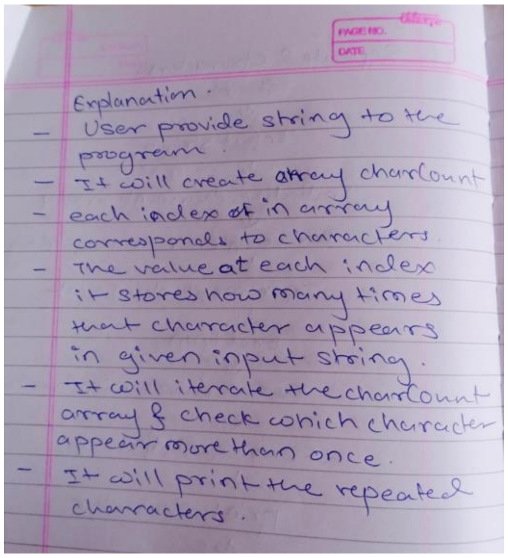


```
C:\Users\CSH\Desktop\ADS>javac RepeatedCharacters.java

C:\Users\CSH\Desktop\ADS>java RepeatedCharacters
Input: "programming"
Output: " g m r
C:\Users\CSH\Desktop\ADS>javac RepeatedCharacters.java

C:\Users\CSH\Desktop\ADS>java RepeatedCharacters
Input: "hello"
Output: " l
C:\Users\CSH\Desktop\ADS>
```

Flowchart-



Explanation –



PAGE NO.
DATE

Explanation.
- User provide string to the program.
- It will create array charCount
- each index ~~of~~ in array corresponds to characters.
- The value at each index it stores how many times that character appears in given input string.
- It will iterate the charCount array & check which character appear more than once.
- It will print the repeated characters.

8. First Non-Repeated Character
Problem: Write a Java program to find the first non-repeated character in a string.

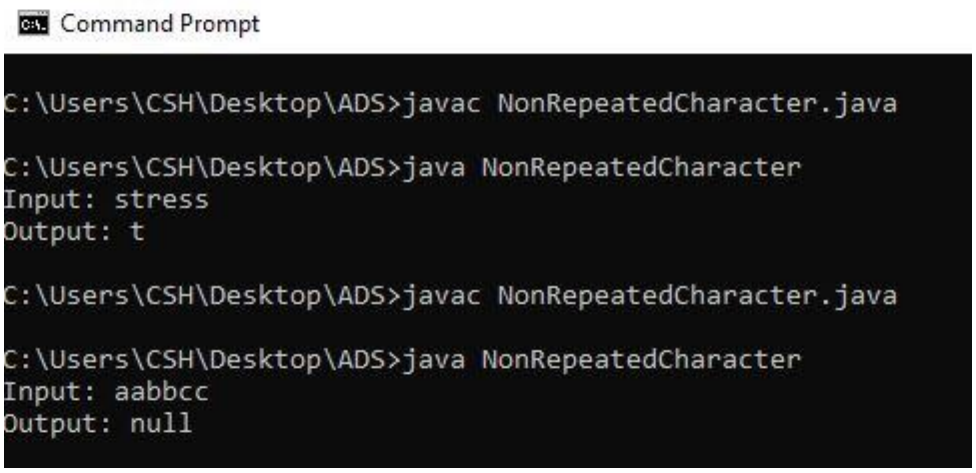
Test Cases:

Input: "stress"
Output: 't'
Input: "aabbcc"
Output: null

Code –

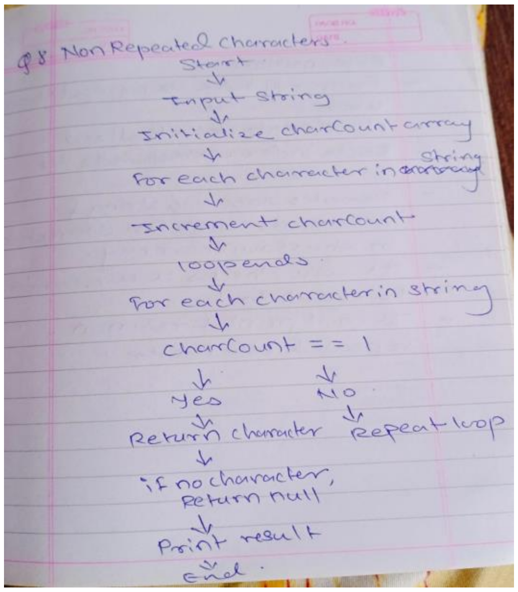
```
import java.util.Scanner;



public class NonRepeatedCharacter {



    public static Character findFirstNonRepeatedCharacter(String input) {

        int[] charCount = new int[256];
        for (int i = 0; i < input.length(); i++) {
            charCount[input.charAt(i)]++;
        }

        for (int i = 0; i < input.length(); i++) {
            if (charCount[input.charAt(i)] == 1) {
                return input.charAt(i);
            }
        }

        return null;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Input: ");
        String input = scanner.nextLine();

        Character result = findFirstNonRepeatedCharacter(input);

        if (result != null) {
            System.out.println("Output: " + result);
        } else {
            System.out.println("Output: " +null);
        }


        scanner.close();
    }
}
```

Output –

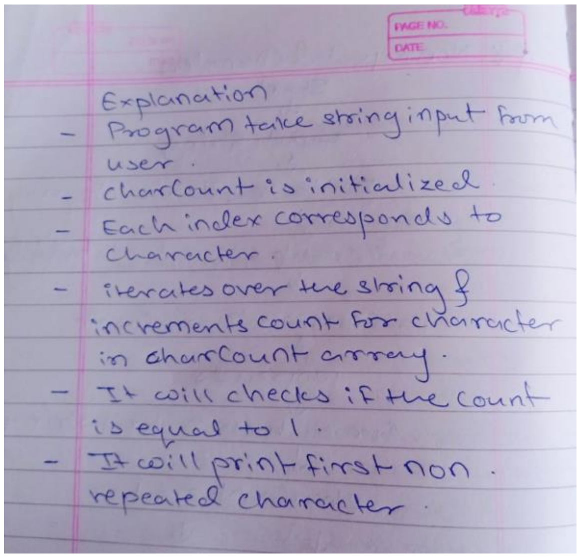


```
Command Prompt
C:\Users\CSH\Desktop\ADS>javac NonRepeatedCharacter.java

C:\Users\CSH\Desktop\ADS>java NonRepeatedCharacter
Input: stress
Output: t

C:\Users\CSH\Desktop\ADS>javac NonRepeatedCharacter.java

C:\Users\CSH\Desktop\ADS>java NonRepeatedCharacter
Input: aabbcc
Output: null
```

Flowchart-


Q8 Non Repeated Characters.
Start
Input String
Initialize charCount array
For each character in ~~[illegible]~~ string
Increment charCount
loop ends
For each character in string
charCount == 1
Yes
No
Return character
Repeat loop
if no character, Return null
Print result
End.

Explanation –


PAGE NO.
DATE

Explanation
- Program take string input from user.
- charCount is initialized.
- Each index corresponds to character.
- iterates over the string & increments count for character in charCount array.
- It will checks if the count is equal to 1.
- It will print first non repeated character.

9. Integer Palindrome
Problem: Write a Java program to check if a given integer is a palindrome.

Test Cases:

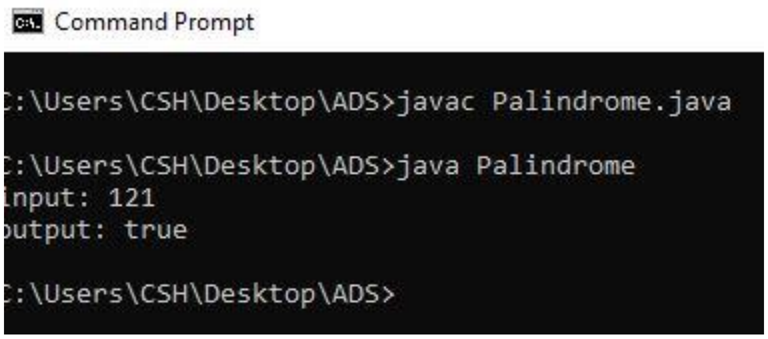
Input: 121
Output: true
Input: -121
Output: false

Code –

```
import java.util.Scanner;



public class Palindrome {



    public static boolean isPalindrome(int number) {
```

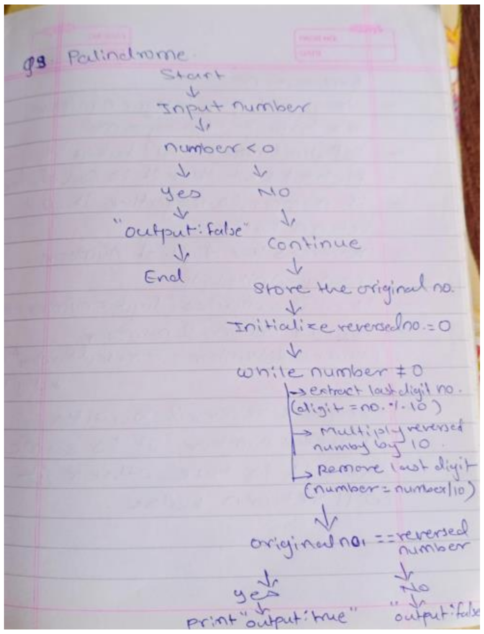
```
    if (number < 0) {
        return false;
    }

    int originalNumber = number;
    int reversedNumber = 0;

    while (number != 0) {
        int digit = number % 10;
        reversedNumber = reversedNumber * 10 + digit;
        number /= 10;
    }

    return originalNumber == reversedNumber;
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.print("input: ");
    int number = scanner.nextInt();

    if (isPalindrome(number)) {
        System.out.println("output: true");
    } else {
            System.out.println("output: false");
        }


        scanner.close();
    }
}
```

Output –

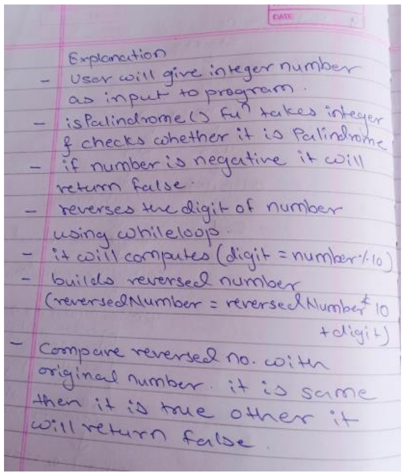


```
:\Users\CSH\Desktop\ADS>javac Palindrome.java

:\Users\CSH\Desktop\ADS>java Palindrome
nput: 121
utput: true

:\Users\CSH\Desktop\ADS>
```

Flowchart-



Explanation –


DATE

Explanation
- User will give integer number as input to program.
- isPalindrome() fu^n takes integer & checks whether it is Palindrome
- if number is negative it will return false.
- reverses the digit of number using whileloop.
- it will computes (digit = number % 10)
- builds reversed number (reversedNumber = reversedNumber * 10 + digit)
- Compare reversed no. with original number. it is same then it is true other it will return false.

10. Leap Year
Problem: Write a Java program to check if a given year is a leap year.

Test Cases:

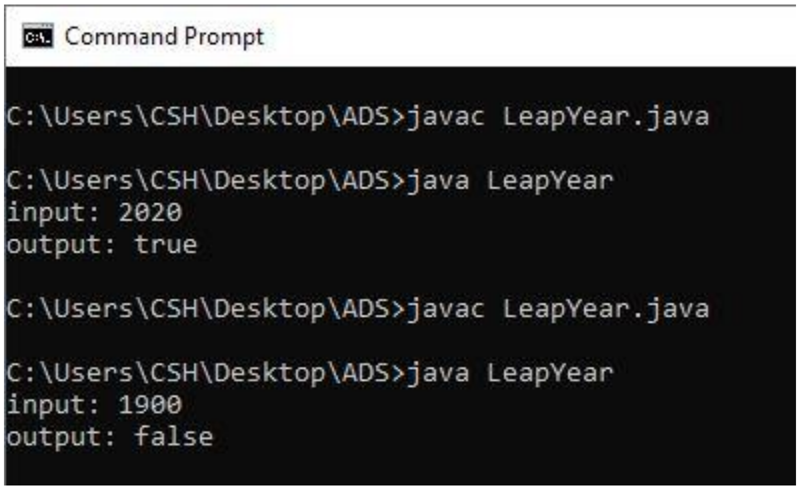
Input: 2020
Output: true
Input: 1900
Output: false

Code –

```
import java.util.Scanner;



public class LeapYear {



    public static boolean isLeapYear(int year) {



        if (year % 4 == 0) {

            if (year % 100 == 0) {

                return year % 400 == 0;

            } else {
```

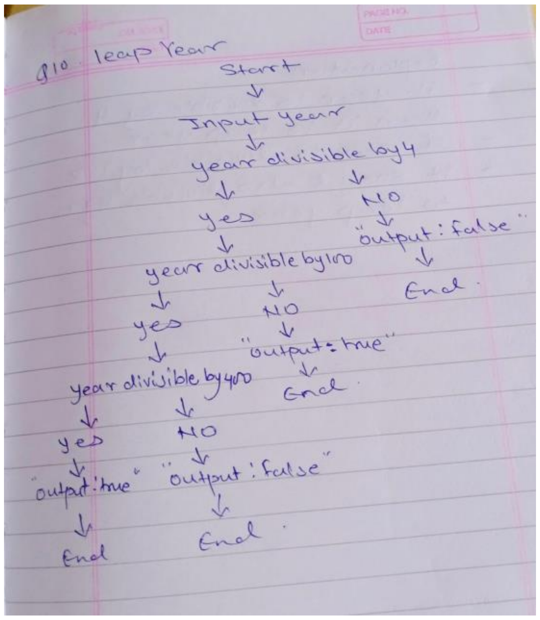
```
            return true;
          }
        }
        return false;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("input: ");
        int year = scanner.nextInt();

        if (isLeapYear(year)) {
            System.out.println("output: true");
        } else {
            System.out.println("output: false");
        }

        scanner.close();
    }
}
```

Output-

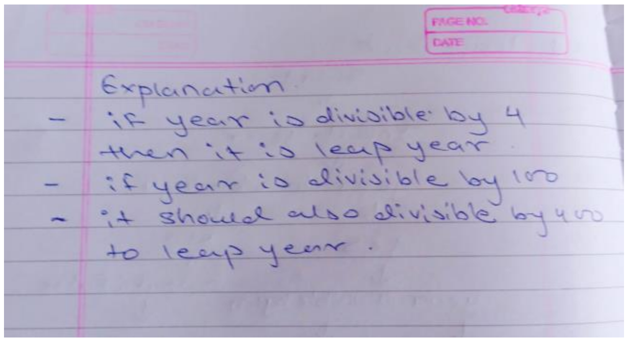


```
C:\Users\CSH\Desktop\ADS>javac LeapYear.java

C:\Users\CSH\Desktop\ADS>java LeapYear
input: 2020
output: true

C:\Users\CSH\Desktop\ADS>javac LeapYear.java

C:\Users\CSH\Desktop\ADS>java LeapYear
input: 1900
output: false
```

Flowchart-



Explanation –



Explanation

- if year is divisible by 4 then it is leap year
- if year is divisible by 100
- it should also divisible by 400 to leap year.