1. **Write a LeapYear program that takes a year as input and outputs the Year is a Leap Year or not a Leap Year.**

Hint =>

a. The LeapYear program only works for year >= 1582, corresponding to a year in the Gregorian calendar. So ensure to check for the same.

a. Further, the Leap Year is a Year divisible by 4 and not 100 unless it is divisible by 400. E.g. 1800 is not a Leap Year and 2000 is a Leap Year.

Write code having multiple *if else* statements based on conditions provided above and a second part having only one if statement and multiple logical

```
import java.util.Scanner;


public class LeapYear {
    public static void main(String[] args) {
        Scanner myobj = new Scanner(System.in);
        int year;


        System.out.println("Enter a year (>= 1582): ");
        year = myobj.nextInt();


        if (year < 1582) {
            System.out.println("Please enter a year greater than or equal to 1582.");
        } else {
            // Approach 1: Using multiple if-else statements
            if (year % 4 == 0) {
                if (year % 100 == 0) {
                    if (year % 400 == 0) {
                        System.out.println(year + " is a Leap Year.");
                    } else {
                        System.out.println(year + " is not a Leap Year.");
```

```
                }
            } else {
                System.out.println(year + " is a Leap Year.");
            }
        } else {
            System.out.println(year + " is not a Leap Year.");
        }

        // Approach 2: Using a single if statement with logical conditions
        if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
            System.out.println("(Using single if) " + year + " is a Leap Year.");
        } else {
            System.out.println("(Using single if) " + year + " is not a Leap Year.");
        }
    }

    myobj.close();
  }
}
```

2. **Rewrite program 1 to determine Leap Year with single if condition using logical and *&&* and or || operators**

```
import java.util.Scanner;

public class LeapYear {
  public static void main(String[] args) {
    Scanner myobj = new Scanner(System.in);
```

```
        int year;

        System.out.println("Enter a year (>= 1582): ");
        year = myobj.nextInt();

        if (year >= 1582) {
            if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
                System.out.println(year + " is a Leap Year.");
            } else {
                System.out.println(year + " is not a Leap Year.");
            }
        } else {
            System.out.println("Please enter a year greater than or equal to 1582.");
        }

        myobj.close();
    }
}
```

3. **Write a program to input marks and 3 subjects physics, chemistry and maths. Compute the percentage and then calculate the grade as per the following guidelines.**

Grade	Remarks	Marks
A	(Level 4, above agency-normalized standards)	80% and above
B	(Level 3, at agency-normalized standards)	70-79%
C	(Level 2, below, but approaching agency-normalized standards)	60-69%
D	(Level 1, well below agency-normalized standards)	50-59%
E	(Level 1- , too below agency-normalized standards)	40-49%
R	(Remedial standards)	39% and below

Hint =>

a. Ensure the Output clearly shows the Average Mark as well as the Grade and Remarks

```
import java.util.Scanner;


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


        System.out.print("Enter Physics marks: ");
        int physics = input.nextInt();


        System.out.print("Enter Chemistry marks: ");
        int chemistry = input.nextInt();


        System.out.print("Enter Maths marks: ");
        int maths = input.nextInt();


        double percentage = (physics + chemistry + maths) / 3.0;
        String grade, remarks;


        if (percentage >= 80) {
            grade = "A";
            remarks = "Level 4, above agency-normalized standards";
        } else if (percentage >= 70) {
            grade = "B";
            remarks = "Level 3, at agency-normalized standards";
        } else if (percentage >= 60) {
```

```
            grade = "C";
            remarks = "Level 2, below, but approaching agency-normalized standards";
        } else if (percentage >= 50) {
            grade = "D";
            remarks = "Level 1, well below agency-normalized standards";
        } else if (percentage >= 40) {
            grade = "E";
            remarks = "Level 1-, too below agency-normalized standards";
        } else {
            grade = "R";
            remarks = "Remedial standards";
        }
         System.out.println("\n--- Student Report ---");
        System.out.println("Average Percentage: " + percentage + "%");
        System.out.println("Grade: " + grade);
        System.out.println("Remarks: " + remarks);

        input.close();
    }
}
```

4. **Write a Program to check if the given number is a prime number or not**

Hint =>

a. A number that can be divided exactly only by itself and 1 are Prime Numbers,

a. Prime Numbers checks are done for numbers greater than 1

a. Loop through all the numbers from 2 to the user input number and check if the reminder is zero. If the reminder is zero break out from the loop as the number is divisible by some other number and is not a prime number.

a. Use the isPrime boolean variable to store the result

```
import java.util.Scanner;

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

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

        boolean isPrime = true;

        if (number <= 1) {
            isPrime = false;
        } else {
            for (int i = 2; i <= Math.sqrt(number); i++) {
                if (number % i == 0) {
                    isPrime = false;
                    break;
                }
            }
        }

        if (isPrime) {
            System.out.println(number + " is a Prime Number.");
        } else {
            System.out.println(number + " is NOT a Prime Number.");
        }

        input.close();
```

```
    }
}
```

5. **Create a program to check if a number is armstrong or not. Use the hints to show the steps clearly in the code**

Hint =>

a. Armstrong Number is a number whose Sum of cubes of each digit results in the original number as in for e.g. 153 = 1^3 + 5^3 + 3^3

a. Get an integer input and store it in the number variable and define sum variable, initialize it to zero and originalNumber variable and assign it to input number variable

a. Use the *while* loop till the originalNumber is not equal to zero

a. In the *while* loop find each digit which is the reminder of the modulus operation *number % 10*. Find the cube of the number and add it to the *sum* variable

a. Again in while loop find the quotient of the number using the division operation *number/10* and assign it to the original number. This removes the last digit of the original number.

a. Finally check if the number and the sum are the same, if same its an Armstrong number else not. So display accordingly

```
import java.util.Scanner;


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


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


        int originalNumber = number;
```

```
        int sum = 0;

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

        if (sum == number) {
            System.out.println(number + " is an Armstrong Number.");
        } else {
            System.out.println(number + " is NOT an Armstrong Number.");
        }

        input.close();
    }
}
```

6. **Create a program to count the number of digits in an integer.**

Hint =>

a. **Get an integer input for the number variable.**

a. **Create an integer variable count with value 0.**

a. **Use a loop to iterate until number is not equal to 0.**

a. **Remove the last digit from number in each iteration**

a. **Increase count by 1 in each iteration.**

a. **Finally display the count to show the number of digits**

```
import java.util.Scanner;
```

```
public class CountDigits {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print("Enter an integer: ");
        int number = input.nextInt();

        int count = 0;
        int originalNumber = number;

        if (number == 0) {
            count = 1;
        } else {
            while (number != 0) {
                number /= 10;
                count++;
            }
        }

        System.out.println("The number " + originalNumber + " has " + count + "
digits.");

        input.close();
    }
}
```

7. **Create a program to find the BMI of a person**

Hint =>

a. Take user input in double for the weight (in kg) of the person and height (in cm) for the person and store it in the corresponding variable.

a. Use the formula BMI = weight / (height * height). Note unit is kg/m^2. For this convert cm to meter

a. Use the table to determine the weight status of the person

```
import java.util.Scanner;


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


        System.out.print("Enter your weight in kg: ");
        double weight = input.nextDouble();


        System.out.print("Enter your height in cm: ");
        double heightCm = input.nextDouble();


        double heightM = heightCm / 100;
        double bmi = weight / (heightM * heightM);


        System.out.printf("Your BMI is: %.2f\n", bmi);


        if (bmi <= 18.4) {
            System.out.println("Status: Underweight");
        } else if (bmi >= 18.5 && bmi <= 24.9) {
            System.out.println("Status: Normal");
        } else if (bmi >= 25.0 && bmi <= 39.9) {
            System.out.println("Status: Overweight");
        } else {
```

```
            System.out.println("Status: Obese");
        }


        input.close();
    }
}
```

8. **Create a program to check if a number taken from the user is a Harshad Number.**

Hint =>

a. A Harshad number is an integer which is divisible by the sum of its digits.

For example, 21 which is perfectly divided by 3 (sum of digits: 2 + 1).

   b. **Get an integer input for the number variable.**
   b. **Create an integer variable sum with initial value 0.**
   c. **Create a while loop to access each digit of the number.**
   d. **Inside the loop, add each digit of the number to sum.**
   e. **Check if the number is perfectly divisible by the sum.**
   f. **If the number is divisible by the sum, print Harshad Number. Otherwise, print Not a Harshad Number.**

```
import java.util.Scanner;
public class HarshadNumberChecker {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);


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

```
        int originalNumber = number;
        int sum = 0;

        while (number > 0) {
            sum += number % 10;
            number /= 10;
        }
        if (originalNumber % sum == 0) {
            System.out.println(originalNumber + " is a Harshad Number.");
        } else {
            System.out.println(originalNumber + " is Not a Harshad Number.");
        }
      input.close();
    }
}
```

9. **Create a program to check if a number is an Abundant Number.**

Hint =>

a. An abundant number is an integer in which the sum of all the divisors of the number is greater than the number itself. For example,

Divisor of 12: 1, 2, 3, 4, 6

Sum of divisor: 1 + 2 + 3 + 4 + 6 = 16 > 12

- **b. Get an integer input for the number variable.**
- **b. Create an integer variable sum with initial value 0.**
- **b. Run a for loop from i = 1 to i < number.**
- **b. Inside the loop, check if number is divisible by i.**
- **b. If true, add i to sum.**
- **b. Outside the loop Check if sum is greater than number.**
- **b. If the sum is greater than the number, print Abundant Number. Otherwise, print Not an Abundant Number.**

```
import java.util.Scanner;

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

        System.out.print("Enter a number: ");
        int number = input.nextInt();
        int sum = 0;

        for (int i = 1; i < number; i++) {
            if (number % i == 0) {
                sum += i;
            }
        }

        if (sum > number) {
            System.out.println(number + " is an Abundant Number.");
        } else {
            System.out.println(number + " is Not an Abundant Number.");
        }

        input.close();
    }
}
```

10. **Write a program to create a calculator using *switch...case*.**

Hint =>

a. Create two double variables named first and second and a String variable named op.

a. Get input values for all variables.

a. The input for the operator can only be one of the four values: "+", "-", "*" or "/".

a. Run a for loop from i = 1 to i < number.

a. Based on the input value of the op, perform specific operations using the *switch...case* statement and print the result.

a. If op is +, perform addition between first and second; if it is -, perform subtraction and so on.

a. If op is neither of those 4 values, print Invalid Operator.

```
import java.util.Scanner;


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


        System.out.print("Enter first number: ");
        double first = input.nextDouble();


        System.out.print("Enter an operator (+, -, *, /): ");
        String op = input.next();


        System.out.print("Enter second number: ");
        double second = input.nextDouble();


        switch (op) {
            case "+":
                System.out.println("Result: " + (first + second));
```

```
            break;
        case "-":
            System.out.println("Result: " + (first - second));
            break;
        case "*":
            System.out.println("Result: " + (first * second));
            break;
        case "/":
            if (second != 0) {
                System.out.println("Result: " + (first / second));
            } else {
                System.out.println("Error: Division by zero is not allowed.");
            }
            break;
        default:
            System.out.println("Invalid Operator.");
    }

    input.close();
  }
}
```

11. **Write a program *DayOfWeek* that takes a date as input and prints the day of the week that the date falls on. Your program should take three command-line arguments: m (month), d (day), and y (year). For m use 1 for January, 2 for February, and so forth. For output print 0 for Sunday, 1 for Monday, 2 for Tuesday, and so forth. Use the following formulas, for the Gregorian calendar (where / denotes integer division):**

$$y_0 = y - (14 - m) / 12$$

$$x = y_0 + y_0/4 - y_0/100 + y_0/400$$

$$m_0 = m + 12 \times ((14 - m) / 12) - 2$$

$$d_0 = (d + x + 31m_0 / 12) \bmod 7$$

```
import java.util.Scanner;

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

        System.out.print("Enter month (1-12): ");
        int m = input.nextInt();

        System.out.print("Enter day (1-31): ");
        int d = input.nextInt();

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

        int y0 = y - (14 - m) / 12;
        int x = y0 + y0 / 4 - y0 / 100 + y0 / 400;
        int m0 = m + 12 * ((14 - m) / 12) - 2;
        int d0 = (d + x + (31 * m0) / 12) % 7;
```

```
        System.out.println("Day of the week (0=Sunday, 1=Monday, ..., 6=Saturday): " +
d0);


        input.close();
    }
}
```