Level 3 Practice Programs

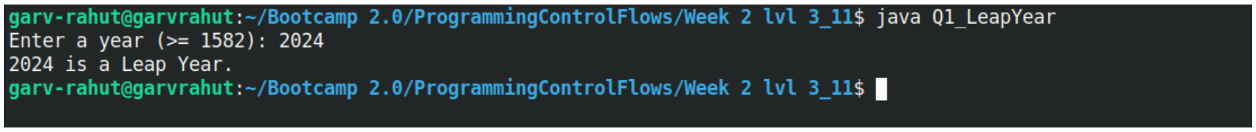
1. Write a LeapYear program that takes a year as input and outputs the Year is a Leap Year or not a Leap Year.

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

```java
import java.util.Scanner;


public class Q1_LeapYear {


  public static void main(String[] args)

  {

      Scanner input = new Scanner(System.in);


      System.out.print("Enter a year (>= 1582): ");


      if (!input.hasNextInt()) {

          System.out.println("Invalid input. Please enter an integer.");

          input.close();

          return;

      }


      int year = input.nextInt();


      if (year < 1582) {
            System.out.println("Leap year calculation only applies to years
>= 1582.");
            input.close();
            return;
        }

        // Part 1: Multiple if-else statements
        //System.out.println("\nUsing 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.");
        }

        /* Part 2: Single if statement with logical operators
        System.out.println("\nUsing a single if statement with logical
operators:");
        if ((year % 4 == 0 && year % 100!= 0) || (year % 400 == 0)) {
            System.out.println(year + */

            input.close();
```

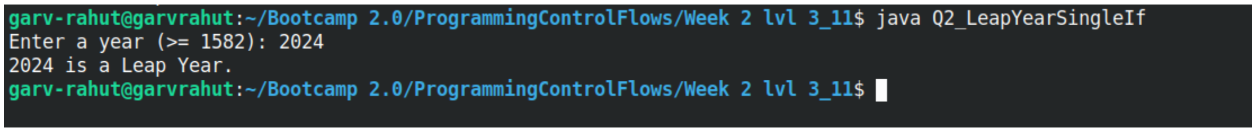
```
    }

}
```



```
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$ java Q1_LeapYear
Enter a year (>= 1582): 2024
2024 is a Leap Year.
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$
```

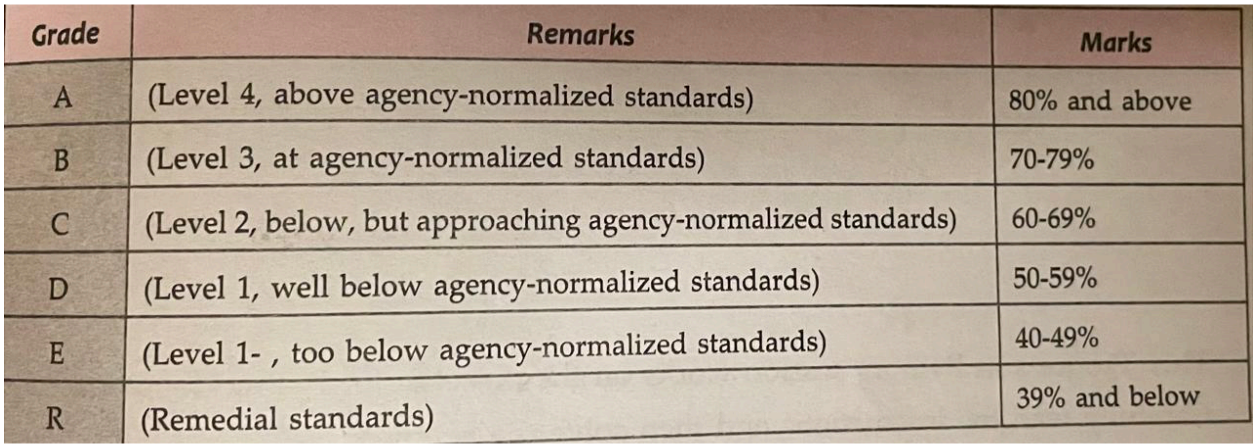
2. Rewrite program 1 to determine Leap Year with single if condition using logical and **&&** and or **||** operators

```
import java.util.Scanner;


public class Q2_LeapYearSingleIf {


    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);


        System.out.print("Enter a year (>= 1582): ");


        if (!input.hasNextInt()) {

            System.out.println("Invalid input. Please enter an integer.");

            input.close();

            return;

        }


        int year = input.nextInt();
        if (year < 1582) {

            System.out.println("Leap year calculation only applies to years >= 1582.");

            input.close();

            return;

        }


        // Single if statement with logical operators

        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.");

        }


        input.close();

    }

}
```



```
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$ java Q2_LeapYearSingleIf
Enter a year (>= 1582): 2024
2024 is a Leap Year.
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$ 
```

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

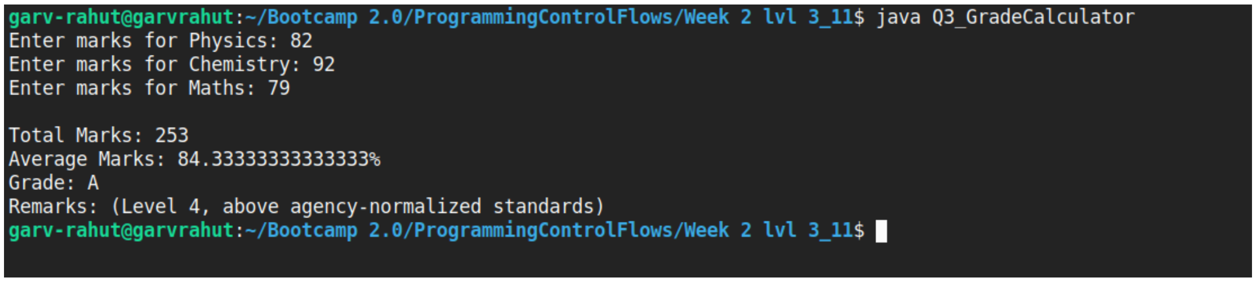


Grade	Remarks	Marks
A	(Level 4, above agency-normalized standards)	80% and above
B	(Level 3, at agency-normalized standards)	70-79%
C	(Level 2, below, but approaching agency-normalized standards)	60-69%
D	(Level 1, well below agency-normalized standards)	50-59%
E	(Level 1- , too below agency-normalized 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 Q3_GradeCalculator {


    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);


        System.out.print("Enter marks for Physics: ");

        int physicsMarks = input.nextInt();


        System.out.print("Enter marks for Chemistry: ");

        int chemistryMarks = input.nextInt();


        System.out.print("Enter marks for Maths: ");

        int mathsMarks = input.nextInt();


        int totalMarks = physicsMarks + chemistryMarks + mathsMarks;

        double percentage = (double) totalMarks / 3;
        System.out.println("\nTotal Marks: " + totalMarks);
        System.out.println("Average Marks: " + percentage + "%");

        String grade;
        String 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("Grade: " + grade);
        System.out.println("Remarks: " + remarks);


        input.close();

    }

}
```



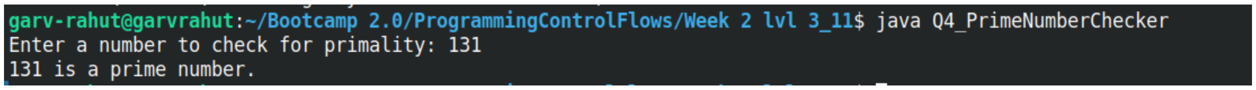
```
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$ java Q3_GradeCalculator
Enter marks for Physics: 82
Enter marks for Chemistry: 92
Enter marks for Maths: 79

Total Marks: 253
Average Marks: 84.33333333333333%
Grade: A
Remarks: (Level 4, above agency-normalized standards)
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$
```

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,
   b. Prime Numbers checks are done for number greater than 1
   c. 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.
   d. Use isPrime boolean variable to store the result

```
import java.util.Scanner;


public class Q4_PrimeNumberChecker {


    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);


        System.out.print("Enter a number to check for primality: ");


        if (!input.hasNextInt()) {
            System.out.println("Invalid input. Please enter an integer.");
            input.close();
            return;
        }


        int number = input.nextInt();

        if (number <= 1) {
            System.out.println("A number must be greater than 1 to be
prime.");
            input.close();
            return;
        }

        boolean isPrime = true; // Assume it's prime initially

        for (int i = 2; i <= Math.sqrt(number); i++) { // Optimized loop:
Check up to sqrt(n)
            if (number % i == 0) {
                isPrime = false; // Found a divisor, so it's not prime
                break; // Exit the loop - no need to check further
            }
        }

        if (isPrime) {
            System.out.println(number + " is a prime number.");
        } else {
            System.out.println(number + " is not a prime number.");
        }


        input.close();

    }

}
```



```
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$ java Q4_PrimeNumberChecker
Enter a number to check for primality: 131
131 is a prime number.
```

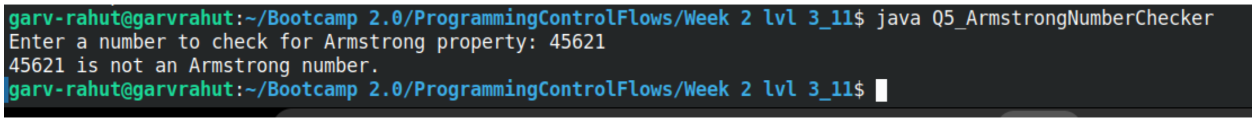
5. Create a program to check if a number is armstrong or not. Use the hints to show the steps clearly in the code

   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
   b. 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
   c. Use the ***while*** loop till the originalNumber is not equal to zero
   d. In the ***while*** loop find the reminder number by using the modulus operator as in ***number % 10***. Find the cube of the number and add it to the ***sum*** variable
   e. Again in while loop find the quotient of the number and assign it to the original number using number / 10 expression. This romoves the last digit of the original number.
   f. 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 Q5_ArmstrongNumberChecker {


    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);


        System.out.print("Enter a number to check for Armstrong property:
");


        if (!input.hasNextInt()) {
            System.out.println("Invalid input. Please enter an integer.");
            input.close();
            return;
        }

        int number = input.nextInt();

        if (number < 0) {
            System.out.println("Please enter a non-negative integer.");
            input.close();
            return;
        }

        int originalNumber = number; // Store the original number
        int sum = 0; // Initialize sum of cubes to 0

        while (originalNumber != 0) {
            int digit = originalNumber % 10; // Get the last digit
(remainder)
            sum += Math.pow(digit, 3); // Calculate cube and add to sum
            originalNumber /= 10; // Remove the last digit (quotient)
        }

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

```java
        input.close();
    }
}
```

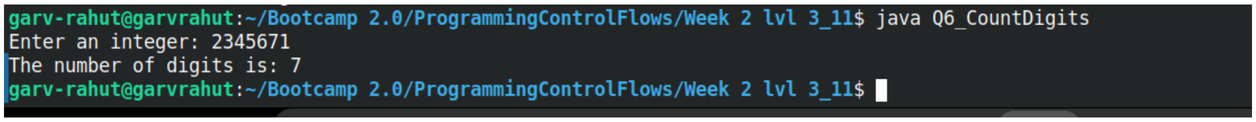


```
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$ java Q5_ArmstrongNumberChecker
Enter a number to check for Armstrong property: 45621
45621 is not an Armstrong number.
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$
```

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

   Hint =>
   a. Get an integer input for the number variable.
   b. Create an integer variable count with value 0.
   c. Use a loop to iterate until number is not equal to 0.
   d. Remove the last digit from number in each iteration
   e. Increase count by 1 in each iteration.
   f. Finally display the count to show the number of digits

```java
import java.util.Scanner;

public class Q6_CountDigits {

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

        System.out.print("Enter an integer: ");

        if (!input.hasNextInt()) {
            System.out.println("Invalid input. Please enter an integer.");
            input.close();
            return;
        }
        int number = input.nextInt();

        if (number == 0) { // Handle the case where the number is 0
separately
            System.out.println("The number of digits is 1.");
            input.close();
            return;
        }

        if (number < 0) {
            number = -number; // Make the number positive if it's negative
        }

        int count = 0;

        while (number != 0) {
            number /= 10; // Remove the last digit
            count++;
        }

        System.out.println("The number of digits is: " + count);

        input.close();
    }
}
```

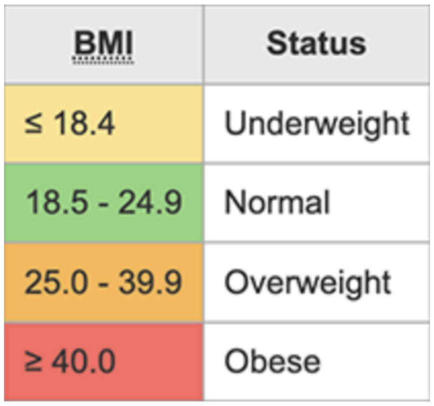


```
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$ java Q6_CountDigits
Enter an integer: 2345671
The number of digits is: 7
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$
```

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

   Hint =>

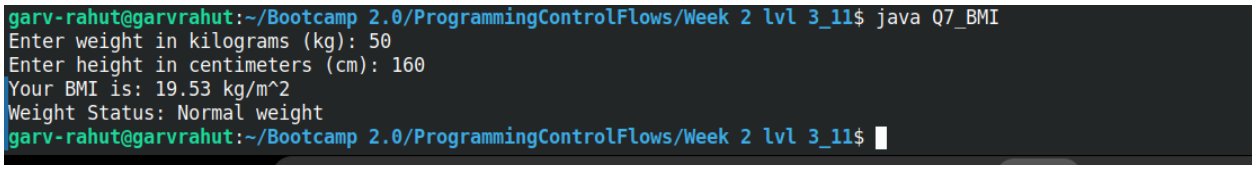
   a. Take user input in double for the weight (in kg) of the person and height (in cm) for the person and store it in the corresponding variable.
   b. Use the formula BMI = weight / (height * height). Note unit is kg/m^2. For this convert cm to meter
   c. Use the table to determine the weight status of the person



BMI	Status
≤ 18.4	Underweight
18.5 - 24.9	Normal
25.0 - 39.9	Overweight
≥ 40.0	Obese

```
import java.util.Scanner;

import java.text.DecimalFormat; // Import for formatting BMI to two
decimal places


public class Q7_BMI {


   public static void main(String[] args) {

       Scanner input = new Scanner(System.in);

       DecimalFormat df = new DecimalFormat("0.00"); // Create
DecimalFormat object


       System.out.print("Enter weight in kilograms (kg): ");

```

```java
        if (!input.hasNextDouble()) {
            System.out.println("Invalid input. Please enter a valid weight (in kg).");
            input.close();
            return;
        }
        double weight = input.nextDouble();

        System.out.print("Enter height in centimeters (cm): ");

        if (!input.hasNextDouble()) {
            System.out.println("Invalid input. Please enter a valid height (in cm).");
            input.close();
            return;
        }

        double heightCm = input.nextDouble();

        if (weight <= 0 || heightCm <= 0) {
            System.out.println("Weight and height must be positive values.");
            input.close();
            return;
        }

        double heightM = heightCm / 100; // Convert height from cm to meters
        double bmi = weight / (heightM * heightM);
        System.out.println("Your BMI is: " + df.format(bmi) + " kg/m^2");
// Format BMI to two decimal places

        String weightStatus;

        if (bmi < 18.5) {
            weightStatus = "Underweight";
        } else if (bmi < 25) {
            weightStatus = "Normal weight";
        } else if (bmi < 30) {
            weightStatus = "Overweight";
        } else {
            weightStatus = "Obese";
        }

        System.out.println("Weight Status: " + weightStatus);

        input.close();
    }
}
```



```
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$ java Q7_BMI
Enter weight in kilograms (kg): 50
Enter height in centimeters (cm): 160
Your BMI is: 19.53 kg/m^2
Weight Status: Normal weight
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$
```

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

   Hint =>

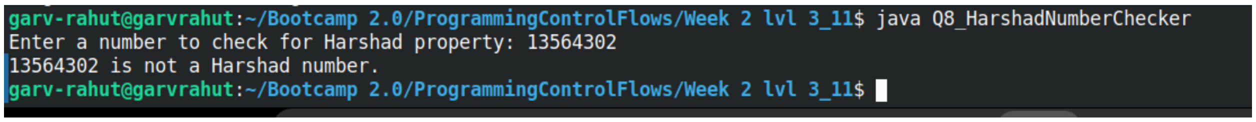
   a. A Harshad number is an integer which is divisible by the sum of its digits.

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

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

```
import java.util.Scanner;

public class Q8_HarshadNumberChecker {

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

       System.out.print("Enter a number to check for Harshad property: ");

       if (!input.hasNextInt()) {
           System.out.println("Invalid input. Please enter an integer.");
           input.close();
           return;
       }

       int number = input.nextInt();

       if (number <= 0) {
           System.out.println("Please enter a positive integer.");
           input.close();
           return;
       }
```

```
        int originalNumber = number; // Store the original number

        int sumOfDigits = 0; // Initialize sum of digits to 0


        while (originalNumber != 0) {

            int digit = originalNumber % 10; // Get the last digit
(remainder)

            sumOfDigits += digit; // Add the digit to the sum

            originalNumber /= 10; // Remove the last digit (quotient)

        }


        if (number % sumOfDigits == 0) {

            System.out.println(number + " is a Harshad number.");

        } else {

            System.out.println(number + " is not a Harshad number.");

        }


        input.close();

    }

}
```



```
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$ java Q8_HarshadNumberChecker
Enter a number to check for Harshad property: 13564302
13564302 is not a Harshad number.
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$ 
```

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

   Hint =>

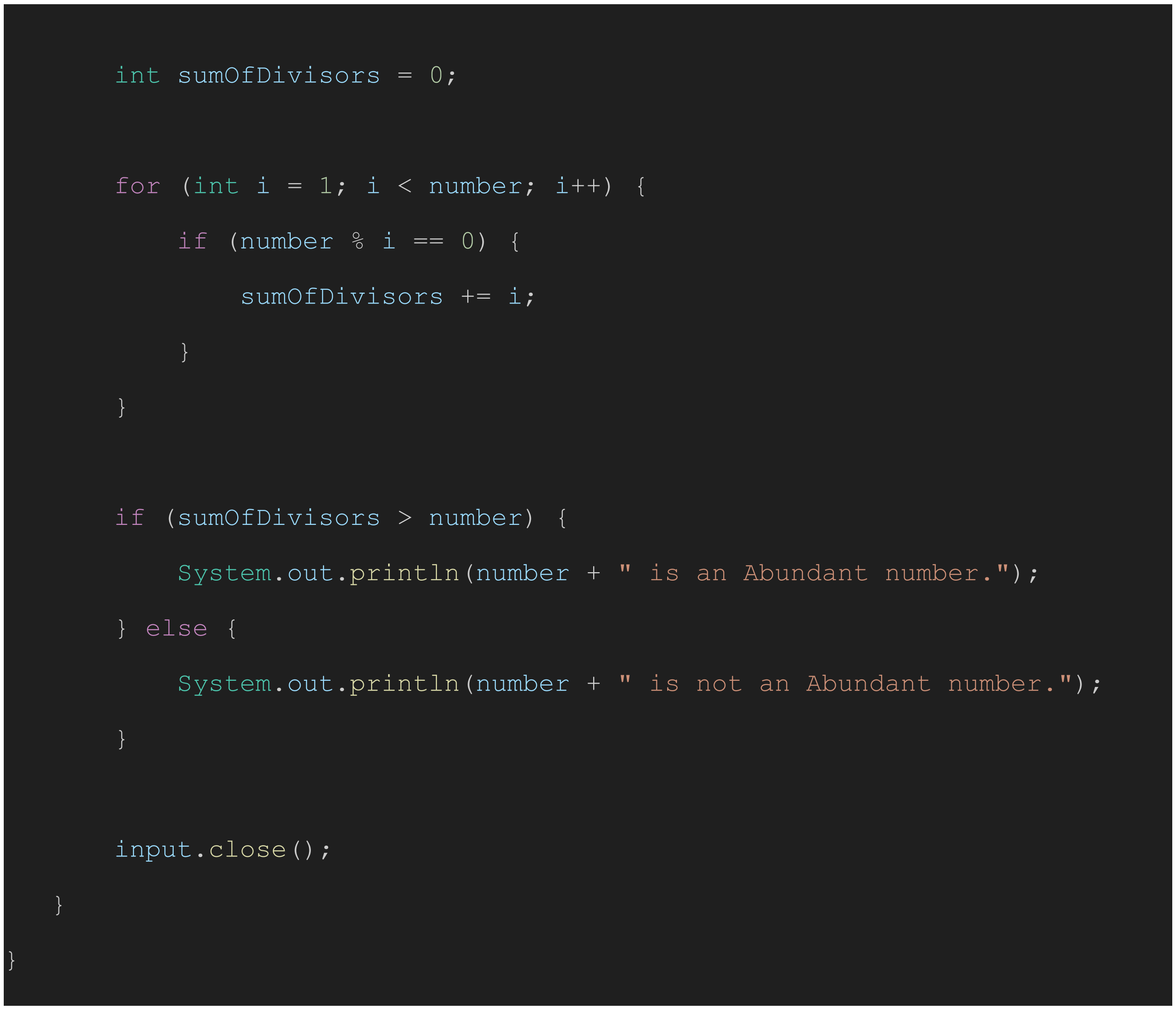
   a. An abundant number is an integer in which the sum of all the divisors of the number is greater than the number itself. For example,

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

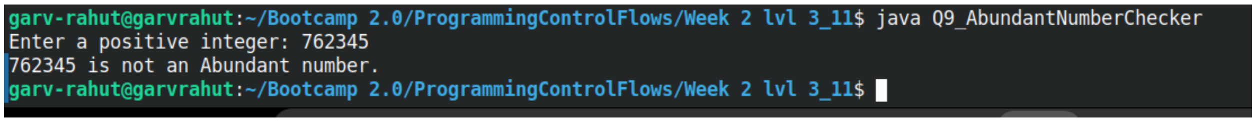
      Sum of divisor: 1 + 2 + 3 + 4 + 6 = 16 > 12

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

```
import java.util.Scanner;

public class Q9_AbundantNumberChecker {

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

       System.out.print("Enter a positive integer: ");

       if (!input.hasNextInt()) {
           System.out.println("Invalid input. Please enter an integer.");
           input.close();
           return;
       }

       int number = input.nextInt();

       if (number <= 0) {
           System.out.println("Please enter a positive integer.");
           input.close();
           return;
       }
```



```
        int sumOfDivisors = 0;

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

        if (sumOfDivisors > number) {
            System.out.println(number + " is an Abundant number.");
        } else {
            System.out.println(number + " is not an Abundant number.");
        }

        input.close();
    }
}
```



```
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$ java Q9_AbundantNumberChecker
Enter a positive integer: 762345
762345 is not an Abundant number.
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$
```

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

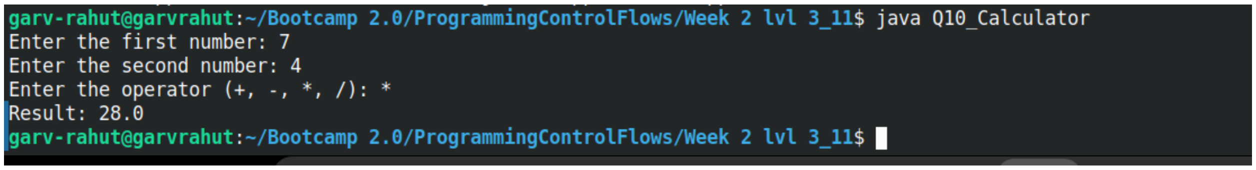
Hint =>

a. Create two double variables named first and second and a String variable named op.
b. Get input values for all variables.
c. The input for the operator can only be one of the four values: "+", "-", "*" or "/".
d. Run a for loop from i = 1 to i < number.
e. Based on the input value of the op, perform specific operations using the ***switch...case*** statement and print the result.

f. If op is +, perform addition between first and second; if it is -, perform subtraction and so on.
g. If op is neither of those 4 values, print Invalid Operator.

```
import java.util.Scanner;

public class Q10_Calculator {

   public static void main(String[] args)

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

       System.out.print("Enter the first number: ");
       if (!input.hasNextDouble()) {
           System.out.println("Invalid input. Please enter a valid
number.");
           input.close();
           return;
       }
       double first = input.nextDouble();

       System.out.print("Enter the second number: ");
       if (!input.hasNextDouble()) {
           System.out.println("Invalid input. Please enter a valid
number.");
           input.close();
           return;
       }
       double second = input.nextDouble();
```

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

double result;

switch (op) {
    case "+":
        result = first + second;
        break;
    case "-":
        result = first - second;
        break;
    case "*":
        result = first * second;
        break;
    case "/":
        if (second == 0) {
            System.out.println("Cannot divide by zero.");
            input.close();
            return;
        }
        result = first / second;
        break;
    default:
        System.out.println("Invalid operator.");
        input.close();
        return; // Exit the program if the operator is invalid
        }


        System.out.println("Result: " + result);


        input.close();

    }

}
```



```
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$ java Q10_Calculator
Enter the first number: 7
Enter the second number: 4
Enter the operator (+, -, *, /): *
Result: 28.0
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$ 
```

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):

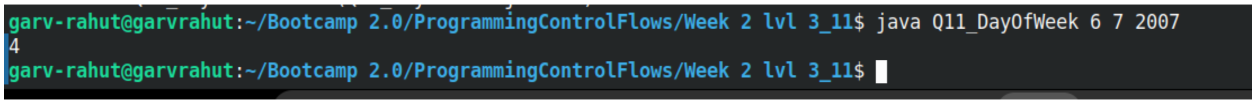
    $y_0 = y - (14 - m) / 12$

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

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

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

```
public class Q11_DayOfWeek {


    public static void main(String[] args) {


        /* if (args.length != 3) {

            System.out.println("Usage: java DayOfWeek m d y");

            return;

        } */

        try {
            int m = Integer.parseInt(args[0]);
            int d = Integer.parseInt(args[1]);
            int y = Integer.parseInt(args[2]);

            if (!isValidDate(m, d, y)) {
                System.out.println("Invalid date.");
                return;
            }

            int dayOfWeek = calculateDayOfWeek(m, d, y);
            System.out.println(dayOfWeek);

        } catch (NumberFormatException e) {
            System.out.println("Invalid input. Month, day, and year must be
integers.");
        }
    }

    private static boolean isValidDate(int m, int d, int y) {
        if (m < 1 || m > 12 || d < 1) {
            return false;
        }

        int[] daysInMonth = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,
31};

        if (m == 2 && isLeapYear(y)) {
            return d <= 29;
        } else {

            return d <= daysInMonth[m];

        }

    }


    private static boolean isLeapYear(int y) {

        return (y % 4 == 0 && y % 100 != 0) || (y % 400 == 0);

    }


    private static int calculateDayOfWeek(int m, int d, int y) {

        int y0 = y - (14 - m) / 12;

        int x = y0 + y0 / 4 - y0 / 100 + y0 / 400;

        int m0 = m + 12 * ((14 - m) / 12) - 2;

        return (d + x + 31 * m0 / 12) % 7;

    }

}
```



```
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$ java Q11_DayOfWeek 6 7 2007
4
garv-rahut@garvrahut:~/Bootcamp 2.0/ProgrammingControlFlows/Week 2 lvl 3_11$ 
```