Index

16	Number Reverser
17	Fascinating Number
18	Bouncy Number
19	Armstrong Number
20	Palindrome Number
21	Complex Addition
22	Pendulum Sort
23	Kaprekar Number
24	Fibonacci Series Sum
25	Insertion Sort

Program - 19

Write a program to input a three digit number. Use a method int Armstrong(int n) to accept the number. The method returns 1, if the number is Armstrong, otherwise zero(0).

Sample Input: 153
Sample Output: 153 ⇒ 13 + 53 + 33 = 153
It is an Armstrong Number.

- Source Code

```java
import java.util.Scanner;

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

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

        boolean isArmstrong = isArmstrongNumber(number);

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

    /**
     * Checks whether a number is an Armstrong number.
     *
     * @param number the number to be checked
     * @return true if the number is an Armstrong number, false otherwise
     */
    public static boolean isArmstrongNumber(int number) {
        int originalNumber = number;  // Stores the original number entered by the user
        int sum = 0;  // Stores the sum of the cubes of the individual digits
        int numDigits = String.valueOf(number).length();  // Stores the number of digits in the number

        while (number > 0) {
            int digit = number % 10;  // Extracts the last digit of the number
            sum += Math.pow(digit, numDigits);  // Adds the cube of the digit to the sum
            number /= 10;  // Removes the last digit from the number
        }

        return sum == originalNumber;  // Returns true if the sum is equal to the original number
    }
}

/**
 * Variable Table:
 *
 * | Variable          | Description                                          |
 * |-------------------|------------------------------------------------------|
 * | `scanner`         | Used to read input from the user                     |
 * | `number`          | Stores the number entered by the user                |
 * | `isArmstrong`     | Stores the result of the check for Armstrong number  |
 * | `originalNumber`  | Stores the original number entered by the user       |
 * | `sum`             | Stores the sum of the cubes of the individual digits|
```

```
 * | `numDigits`       | Stores the number of digits in the number          |
 * | `digit`           | Stores the last digit extracted from the `number`  |
 */
```

- Output

```
Enter a number: 371
371 is an Armstrong number.
```

Program - 13

Write a program in Java to create a 4X4 matrix and store the different numbers using input statements. Arrange the numbers of each row in ascending order and display the results.

- Source Code

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

public class MatrixSorter {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int[][] matrix = new int[4][4];

        // Prompt the user to input the matrix values
        System.out.println("Enter the matrix values:");

        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                matrix[i][j] = scanner.nextInt();
            }
        }

        System.out.println("Original matrix:");
        printMatrix(matrix);

        sortMatrix(matrix);

        System.out.println("\nSorted matrix:");
        printMatrix(matrix);
    }

    /**
     * Sorts the elements in each row of the matrix in ascending order.
     *
     * @param matrix the matrix to be sorted
     */
    public static void sortMatrix(int[][] matrix) {
        for (int i = 0; i < matrix.length; i++) {
            Arrays.sort(matrix[i]);
        }
    }

    /**
     * Prints the elements of the matrix.
     *
     * @param matrix the matrix to be printed
     */
    public static void printMatrix(int[][] matrix) {
        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length; j++) {
                System.out.print(matrix[i][j] + " ");
            }
            System.out.println();
        }
    }
}

/**
 * Variable Table:
 *
```

```
 * | Variable | Description                                          |
 * |----------|------------------------------------------------------|
 * | `scanner`| Used to read input from the user                     |
 * | `matrix` | Stores the matrix entered by the user                |
 * | `i`      | Loop variable for rows in the matrix                 |
 * | `j`      | Loop variable for columns in the matrix              |
 *
 */
```

- Output

```
Enter the matrix values:
14 51 71 124
151 351 12 5
51 623 1 415
14 621 63 1

Original matrix:
14 51 71 124
151 351 12 5
51 623 1 415
14 621 63 1

Sorted matrix:
14 51 71 124
5 12 151 351
1 51 415 623
1 14 63 621
```

Program - 7

Write a program in Java to accept Binary Number (Base 2) and convert it to its Decimal Number (Base 10).

- Source Code

```java
import java.util.Scanner;

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

        System.out.print("Enter a binary number: ");
        String binaryStr = scanner.nextLine();

        int decimal = 0;  // Stores the decimal equivalent of the binary number
        int power = 0;  // Represents the power of 2 for each binary digit

        for (int i = binaryStr.length() - 1; i >= 0; i--) {
            char digit = binaryStr.charAt(i);

            if (digit == '1') {
                decimal += Math.pow(2, power);
            }

            power++;
        }

        System.out.println("Decimal equivalent: " + decimal);

        scanner.close();
    }
}

/**
 * Variable Table:
 *
 * | Variable      | Description                                          |
 * |---------------|------------------------------------------------------|
 * | `scanner`     | Used to read input from the user                     |
 * | `binaryStr`   | Stores the binary number entered by the user         |
 * | `decimal`     | Stores the decimal equivalent of the binary number   |
 * | `power`       | Represents the power of 2 for each binary digit      |
 * | `i`           | Loop variable for iterating through the binary string |
 * | `digit`       | Represents a binary digit (character)                |
 */
```

- Output

```
Enter a binary number: 100
Decimal equivalent: 4
```

Program - 18

A number is said to Bouncy number if the digits of the number are unsorted.

For example,
22344 - It is not a Bouncy number because the digits are sorted in ascending order.
774410 - It is not a Bouncy number because the digits are sorted in descending order.
155349 - It is a Bouncy number because the digits are unsorted.
A number below 100 can never be a Bouncy number.

Write a program in java to accept a number. Check and display whether it is a Bouncy number or not.

- Source Code

```
import java.util.Scanner;

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

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

        boolean isBouncy = isBouncyNumber(number);

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

        scanner.close();
    }

    public static boolean isBouncyNumber(int number) {
        boolean increasing = false;
        boolean decreasing = false;

        int lastDigit = number % 10;
        number /= 10;

        while (number > 0) {
            int currentDigit = number % 10;
            number /= 10;

            if (currentDigit > lastDigit) {
                increasing = true;
            } else if (currentDigit < lastDigit) {
                decreasing = true;
            }

            if (increasing && decreasing) {
                return true;
            }

            lastDigit = currentDigit;
        }
```

```
        return false;
    }
}

/**
 * Variable Table:
 *
 * | Variable       | Description                                      |
 * |----------------|--------------------------------------------------|
 * | `scanner`      | Used to read input from the user                 |
 * | `number`       | Stores the number entered by the user            |
 * | `isBouncy`     | Stores the result of the check for Bouncy number |
 * | `increasing`   | Indicates if the digits are increasing           |
 * | `decreasing`   | Indicates if the digits are decreasing           |
 * | `lastDigit`    | Stores the last digit of the number              |
 * | `currentDigit` | Stores the current digit being compared          |
 */
```

Output

```
Enter a number: 516
516 is a Bouncy number.
```

Program - 2

Write a program in Java to perform the Bubble Sort Algorithm Technique in Array.

- Source Code

```java
public class BubbleSort {
    public static void main(String[] args) {
        int[] arr = {9, 5, 2, 3};
        int n = arr.length;

        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }

        for (int i = 0; i < arr.length; i++) {
            System.out.println(arr[i]);
        }
    }
}

/**
 * Variable Table:
 *
 * | Variable | Description                                                |
 * |----------|------------------------------------------------------------|
 * | `arr`    | Stores the array of integers to be sorted                  |
 * | `n`      | Stores the length of the array                             |
 * | `i`      | Loop variable for the outer loop (controls passes)         |
 * | `j`      | Loop variable for the inner loop (controls comparisons)    |
 * | `temp`   | Temporary variable for swapping elements in the array      |
 */
```

- Output

```
2
3
5
9
```

Program - 8

A Circular Prime is a prime number that remains prime under cyclic shifts of its digits. When the leftmost digit is removed and replaced at the end of the remaining string of digits, the generated number is still prime. The process is repeated until the original number is reached again.

A number is said to be prime if it has only two factors 1 and itself.

- Source Code

```java
import java.util.Scanner;

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

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

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

        scanner.close();
    }

    public static boolean isCircularPrime(int number) {
        if (number < 2 || !isPrime(number)) {
            return false;
        }

        String numStr = Integer.toString(number);

        for (int i = 0; i < numStr.length() - 1; i++) {
            numStr = numStr.substring(1) + numStr.charAt(0);
            if (!isPrime(Integer.parseInt(numStr))) {
                return false;
            }
        }

        return true;
    }

    public static boolean isPrime(int number) {
        if (number < 2) {
            return false;
        }

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

        return true;
    }
}
```

```
/**
 * Variable Table:
 *
 * | Variable   | Description                                    |
 * |------------|------------------------------------------------|
 * | `scanner`  | Used to read input from the user               |
 * | `number`   | Stores the number entered by the user          |
 * | `numStr`   | Stores the string representation of the number |
 * | `i`        | Loop variable for iterating through the number |
 */
```

- Output

```
Enter a number: 341
341 is not a circular prime.
```

Program - 21

Write a program in Java to calculate the addition of two Complex number using passing and returning object concept. Design a class Complex with the following details:
Data Members :
x : stores the real part
y : stores the imaginary part
Member Functions:
Complex (…) : Parameterized constructor to assign value to data members.
void display ( ) : To display the Complex Number.
Complex add (Complex obj) : calculates and returns the sum of the two complex numbers.

- Source Code

```
import java.util.Scanner;

class ComplexNumbers {
    int x, y;

    ComplexNumbers() {x = 0; y = 0;}

    ComplexNumbers(int xx, int yy) {x = xx; y = yy;}

    ComplexNumbers add(ComplexNumbers obj) {
        ComplexNumbers result = new ComplexNumbers();
        result.x = this.x + obj.x;
        result.y = this.y + obj.y;
        return result;
    }

    void display() {
        if (y >= 0)
            System.out.println(x + " + i" + y);
        else
            System.out.println(x + " -i" + Math.abs(y));
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the real and imaginary part: ");
        int x1 = scanner.nextInt();
        int y1 = scanner.nextInt();
        System.out.print("Enter the real and imaginary part: ");
        int x2 = scanner.nextInt();
        int y2 = scanner.nextInt();

        ComplexNumbers c1 = new ComplexNumbers(x1, y1);
        ComplexNumbers c2 = new ComplexNumbers(x2, y2);
        ComplexNumbers c3 = new ComplexNumbers();
        c3 = c1.add(c2);

        System.out.println("First Complex Number:");
        c1.display();
        System.out.println("Second Complex Number:");
        c2.display();
        System.out.println("Final Complex Number:");
        c3.display();
```

```
    }
}

/**
 * Variable Table:
 *
 * | Variable   | Description                                    |
 * |------------|------------------------------------------------|
 * | `x`        | Real part of the complex number                |
 * | `y`        | Imaginary part of the complex number           |
 * | `scanner`  | Used to read input from the user               |
 * | `x1`       | Real part of the first complex number entered  |
 * | `y1`       | Imaginary part of the first complex number entered |
 * | `x2`       | Real part of the second complex number entered |
 * | `y2`       | Imaginary part of the second complex number entered|
 * | `c1`       | Object representing the first complex number   |
 * | `c2`       | Object representing the second complex number  |
 * | `c3`       | Object representing the result of addition     |
 */
```

- Output

```
Enter the real and imaginary part: 3 9
Enter the real and imaginary part: 1 0
First Complex Number:
3 + i9
Second Complex Number:
1 + i0
Final Complex Number:
4 + i9
```

Program - 4

Write a program in Java which will give number of days in a month depending of the month entered by the user.

- Source Code

```
import java.util.Scanner;

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

        int days = 0;
        String monthName = "Unknown";

        System.out.print("Input a month number: ");
        int monthNumber = scanner.nextInt();

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

        switch (monthNumber) {
            case 1:
                monthName = "January";
                days = 31;
                break;
            case 2:
                monthName = "February";
                if ((year % 400 == 0) || ((year % 4 == 0) && (year % 100 != 0))) {
                    days = 29;
                } else {
                    days = 28;
                }
                break;
            case 3:
                monthName = "March";
                days = 31;
                break;
            case 4:
                monthName = "April";
                days = 30;
                break;
            case 5:
                monthName = "May";
                days = 31;
                break;
            case 6:
                monthName = "June";
                days = 30;
                break;
            case 7:
                monthName = "July";
                days = 31;
                break;
            case 8:
                monthName = "August";
                days = 31;
                break;
            case 9:
                monthName = "September";
                days = 30;
```

```
                break;
            case 10:
                monthName = "October";
                days = 31;
                break;
            case 11:
                monthName = "November";
                days = 30;
                break;
            case 12:
                monthName = "December";
                days = 31;
                break;
        }

        System.out.println(monthName + " " + year + " has " + days + " days");
    }
}

/**
 * Variable Table:
 *
 * | Variable      | Description                                    |
 * |---------------|------------------------------------------------|
 * | `days`        | Number of days in the given month              |
 * | `monthName`   | Name of the month corresponding to the number  |
 * | `scanner`     | Used to read input from the user               |
 * | `monthNumber` | Input month number provided by the user        |
 * | `year`        | Input year provided by the user                |
 */
```

- Output

```
Input a month number: 2
Input a year: 2016
February 2016 has 29 days
```

Program - 12

Write a program in Java to store 10 different numbers in a single dimensional array. Display the numbers after eliminating the duplicate numbers from the array

- Source Code

```java
import java.util.Arrays;

public class duplicateDeleter {
    public static void main(String[] args) {
        int[] numbers = { 1, 2, 3, 4, 5, 6, 7, 8, 7, 10 };

        int[] uniqueNumbers = eliminateDuplicates(numbers);
        System.out.println("Final set of numbers: " + Arrays.toString(uniqueNumbers));
    }

    public static int[] eliminateDuplicates(int[] numbers) {
        int[] uniqueNumbers = new int[numbers.length];
        int index = 0;

        for (int i = 0; i < numbers.length; i++) {
            boolean isDuplicate = false;
            for (int j = 0; j < i; j++) {
                if (numbers[i] == numbers[j]) {
                    isDuplicate = true;
                    break;
                }
            }
            if (!isDuplicate) {
                uniqueNumbers[index++] = numbers[i];
            }
        }

        return Arrays.copyOf(uniqueNumbers, index);
    }
}
```

Output

```
Final set of numbers: [1, 2, 3, 4, 5, 6, 7, 8, 10]
```

Program - 17

Write a Program in Java to input a number and check whether it is a Fascinating Number or not.

Fascinating Numbers: Some numbers of 3 digits or more exhibit a very interesting property. The property is such that, when the number is multiplied by 2 and 3, and both these products are concatenated with the original number, all digits from 1 to 9 are present exactly once, regardless of the number of zeroes.

- Source Code

```java
import java.util.HashSet;
import java.util.Scanner;
import java.util.Set;

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

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

        boolean isFascinating = isFascinatingNumber(number);

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

        scanner.close();
    }

    public static boolean isFascinatingNumber(int number) {
        String concatenatedString = String.valueOf(number) +
                String.valueOf(number * 2) +
                String.valueOf(number * 3);

        Set<Character> digitSet = new HashSet<>();

        for (char digit : concatenatedString.toCharArray()) {
            if (digit != '0') {
                if (!digitSet.add(digit)) {
                    return false;
                }
            }
        }

        return digitSet.size() == 9;
    }

    /**
     * Variable Table:
     *
     * | Variable            | Description                                          |
     * |---------------------|------------------------------------------------------|
     * | `number`            | Input number provided by the user                    |
     * | `isFascinating`     | Indicates whether the number is a fascinating number |
     * | `concatenatedString`| String formed by concatenating the number and its multiples |
     * | `digitSet`          | Set to store unique digits encountered in the string |
```

```
     * | `scanner`           | Used to read input from the user                    |
     */
}
```

- Output

```
Enter a number: 451
451 is not a fascinating number.
```

Program - 24

Write a program in Java to find the Fibonacci Numbers up until the value inputted by the user. Give a list of all fibonacci numbers nearest to the user defined value

- Source Code

```java
import java.util.*;

class FibonacciSeries {
    public static int fibo(int n) {
        if (n == 0 || n == 1) { // Base case
            return n;
        } else {
            return (fibo(n - 1) + fibo(n - 2)); // Recursive call
        }
    }

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

        System.out.print("Enter the number: ");
        int n = sc.nextInt();
        int a = 1, b; // Initialize

        while (true) {
            b = fibo(a);

            if (b <= n) { // Checking
                System.out.println(b);
            } else {
                break;
            }

            a++;
        }
    }

    /**
     * Variable Table:
     *
     * | Variable | Description                                    |
     * |----------|------------------------------------------------|
     * | sc       | Scanner object to read user input              |
     * | n        | Input number provided by the user              |
     * | a        | Counter for generating Fibonacci series        |
     * | b        | Current Fibonacci number                       |
     */
}
```

- Output

```
Enter the number: 69
1
1
2
3
5
8
```

```
13
21
34
55
```

Program - 14

A happy number is a number which eventually reaches 1 when replaced by the sum of the square of each digit.

- Source Code

```java
import java.util.HashSet;
import java.util.Scanner;
import java.util.Set;

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

        System.out.print("Enter a number to check if it is a happy number: ");
        int number = sc.nextInt();

        Set<Integer> seen = new HashSet<>();
        boolean isHappy = false;

        while (!seen.contains(number)) {
            seen.add(number);

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

            number = sum;
            if (number == 1) {
                isHappy = true;
                break;
            }
        }

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

    /**
     * Variable Table:
     *
     * | Variable | Description                                          |
     * |----------|------------------------------------------------------|
     * | sc       | Scanner object to read user input                    |
     * | number   | Input number provided by the user                    |
     * | seen     | Set to keep track of previously seen numbers         |
     * | isHappy  | Boolean flag indicating if the number is a happy number|
     * | sum      | Sum of squared digits of the current number          |
     * | digit    | Current digit extracted from the number              |
     */
}
```

- Output

```
Enter a number to check if it is a happy number: 42
42 is not a happy number.
```

Program - 25

Write a program in Java to accept 10 numbers in an array and sort them ascending order using Insertion sort technique. Display the results with proper Message.

- Source Code

```java
import java.util.*;

class InsertionSort {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] a = new int[10]; // Creating array

        for (int i = 0; i < 10; i++) {
            System.out.print("Enter a number: ");
            a[i] = sc.nextInt(); // To store number in an array
        }

        for (int i = 1; i < 10; i++) {
            int t = a[i];
            int j = i - 1;

            while (j >= 0 && t < a[j]) { // Checking
                a[j + 1] = a[j];
                j--;
            }

            a[j + 1] = t; // Swapping
        }

        System.out.println("After sorting:");
        for (int i = 0; i < 10; i++)
            System.out.print(a[i] + " ");
    }

    /**
     * Variable Table:
     *
     * | Variable | Description                              |
     * |----------|------------------------------------------|
     * | sc       | Scanner object to read user input        |
     * | a        | Integer array to store the input numbers |
     * | i        | Loop variable for iterating over the array|
     * | t        | Temporary variable for swapping elements |
     * | j        | Loop variable for comparing elements     |
     */
}
```

- Output

```
Enter a number: 5
Enter a number: 1
Enter a number: 3
Enter a number: 2
Enter a number: 7
Enter a number: 3
Enter a number: 4
Enter a number: 2
Enter a number: 7
```

```
Enter a number: 2
After sorting:
1 2 2 2 3 3 4 5 7 7
```

Program - 23

A positive whole number ‘n’ that has ‘d’ number of digits is squared and split into two pieces, a right-hand piece that has ‘d’ digits and a left-hand piece that has remaining ‘d’ or ‘d-1’ digits. If the sum of the two pieces is equal to the number, then ‘n’ is a Kaprekar number. The first few Kaprekar numbers are: 9, 45, 297 ……..

Write a program in Java to display all Kaprekar numbers from 1 to 1000 and also calculate their sum & display the result with a proper message.

- Source Code

```
import java.util.*;
public class KarpekarNumbers {
    static boolean kaprekar(int n) {
        if (n == 1)
            return true;

        int sq_n = n * n;
        int copy = sq_n;
        int d = 0;

        while (copy != 0) {
            d++;
            copy /= 10;
        }

        for (int r_digits = 1; r_digits < d; r_digits++) {
            int eq_parts = (int) Math.pow(10, r_digits);

            if (eq_parts == n)
                continue;

            int sum = sq_n / eq_parts + sq_n % eq_parts;

            if (sum == n)
                return true;
        }

        return false;
    }

    static int s = 0;
    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number to which you want the sum");
        int n = sc.nextInt();
        for (int i = 1; i < n; i++) {
            if (kaprekar(i)) {
                System.out.print(i + " ");
                s = s + i;
            }
        }

        System.out.println();
        System.out.println("Sum of all Karpekar Numbers: " + s);
    }
```

```
/**
 * Variable Table:
 * | Variable  | Description                                                  |
 * |-----------|--------------------------------------------------------------|
 * | n         | Current number being checked                                 |
 * | sq_n      | Square of the current number                                 |
 * | copy      | Copy of the square number to calculate the number of digits  |
 * | d         | Number of digits in the square number                        |
 * | r_digits  | Number of right-side digits to consider for partitioning     |
 * | eq_parts  | Equal parts for partitioning the square number               |
 * | sum       | Sum of the two partitions                                    |
 * | s         | Sum of all Karpekar Numbers from 1 to 1000                   |
 */
```

- Output

```
Enter the number to which you want the sum 1000
1 9 45 55 99 297 703 999
Sum of all Karpekar Numbers: 2208
```

Program - 16

Write a program in Java to input a number and reverse the digits of the number by using recursive function. Display the new number.

- Source Code

```java
import java.util.Scanner;

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

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

        int reversedNumber = reverseNumber(number);

        System.out.println("Reversed number: " + reversedNumber);
    }

    public static int reverseNumber(int number) {
        if (number < 10) {
            return number;
        }

        int lastDigit = number % 10;
        int remainingDigits = number / 10;

        int reversed = reverseNumber(remainingDigits);
        int numDigits = (int) Math.log10(remainingDigits) + 1;

        return lastDigit * (int) Math.pow(10, numDigits) + reversed;
    }

    /**
     * Variable Table:
     *
     * | Variable          | Description                                      |
     * |-------------------|--------------------------------------------------|
     * | sc                | Used to read input from the user                 |
     * | number            | Input number provided by the user                |
     * | reversedNumber    | Reversed number of the input number              |
     * | lastDigit         | Last digit of the number                         |
     * | remainingDigits   | Number obtained after removing the last digit    |
     * | reversed          | Reversed number of the remaining digits          |
     * | numDigits         | Number of digits in the remaining digits         |
     */
}
```

- Output

```
Enter a number: 5215
Reversed number: 5125
```

Program - 20

Palindrome Number in Java: Write a program to accept a number from the user and check whether it is a Palindrome number or not. A number is a Palindrome which when reads in reverse order is same as in the right order.

- Source Code

```java
import java.util.Scanner;

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

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

        boolean isPalindrome = isPalindromeNumber(number);

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

        scanner.close();
    }

    public static boolean isPalindromeNumber(int number) {
        int originalNumber = number;
        int reversedNumber = 0;

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

        return originalNumber == reversedNumber;
    }

    /**
     * Variable Table:
     *
     * | Variable          | Description                                    |
     * |-------------------|------------------------------------------------|
     * | scanner           | Used to read input from the user               |
     * | number            | Input number provided by the user              |
     * | isPalindrome      | Indicates whether the number is palindrome     |
     * | originalNumber    | Original input number                          |
     * | reversedNumber    | Reversed number of the input number            |
     * | digit             | Current digit extracted from the number        |
     */
}
```

- Output

```
Enter a number: 789987
789987 is a palindrome number.
```

Program - 1

Write a program in Java to print the following pattern.

```
A
AA
AAA
AAAA
AAAAA
```

- Source Code

```java
public class PatternA {
    public static void main(String[] args) {
        for (int i = 0; i <= 5; i++) {
            for (int j = 0; j < i; j++) {
                System.out.print("A");
            }
            System.out.println();
        }
    }

    /**
     * Variable Table:
     *
     * | Variable | Description                                |
     * |----------|--------------------------------------------|
     * | `i`      | Represents the row number in the outer loop |
     * | `j`      | Represents the column number in the inner loop |
     */
}
```

- Output

```
A
AA
AAA
AAAA
AAAAA
```

Program - 22

Write a program to accept a set of n integers (where n > 0) in a single dimensional array. Arrange the elements of the array such that the lowest number appears in the center of the array, next lower number in the right cell of the center, next lower in the left cell of the center and so on… . The process will stop when the highest number will set in its appropriate cell. Finally, display the array elements. (Pendulum Arrangement of Array)

- Source Code

```
import java.util.Scanner;

class PendulumSort {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the order: ");
        int n = sc.nextInt(); // Input the order of matrix from the user

        if (n > 1 && n <= 20) {
            int a[] = new int[n]; // Initializing array
            int b[] = new int[n]; // Initializing array

            for (int i = 0; i < n; i++) {
                System.out.print("Enter number: ");
                a[i] = sc.nextInt(); // Store the number in the array
            }

            System.out.println("Original array:");
            for (int i = 0; i < n; i++)
                System.out.print(a[i] + " ");

            // Sorting
            for (int i = 0; i < n - 1; i++) {
                for (int j = 0; j < n - 1 - i; j++) {
                    if (a[j] < a[j + 1]) { // Checking
                        int t = a[j]; // Swapping
                        a[j] = a[j + 1]; // Swapping
                        a[j + 1] = t; // Swapping
                    }
                }
            }

            int m = n / 2; // To store the middle index
            b[m] = a[0];
            int l = m - 1;
            int r = m + 1;

            for (int i = 1; i < n;) {
                b[r++] = a[i++];
                b[l--] = a[i++];
            }

            System.out.println();
            System.out.println("Rearranged array:");
            for (int i = 0; i < n; i++)
                System.out.print(b[i] + " ");
            System.out.println("                                                    ");
        } else {
            System.out.println("Invalid Range");
        }
```

```
    }

    /**
     * Variable Table:
     *
     * | Variable | Description                                          |
     * |----------|------------------------------------------------------|
     * | `n`      | Represents the order of the matrix                   |
     * | `a`      | Array used to store the numbers                      |
     * | `b`      | Array used to store the rearranged numbers           |
     * | `m`      | Represents the middle index of the array             |
     * | `l`      | Represents the left index for rearranging the numbers|
     * | `r`      | Represents the right index for rearranging the numbers|
     */
}
```

- Output

```
Enter the order: 5
Enter number: 1
Enter number: 61
Enter number: 613
Enter number: 21
Enter number: 5
Original array:
1 61 613 21 5
Rearranged array:
1 21 613 61 5
```

Program - 10

Write a program that encodes a word into Piglatin. To translate word into Piglatin word, convert the word into uppercase and then place the first vowel of the original word as the start of the new word along with the remaining alphabets. The alphabets present before the vowel being shifted towards the end followed by "AY".

- Source Code

```java
import java.util.Scanner;

public class PigLatin {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String inputString = scanner.nextLine();

        String[] words = inputString.split("\\s+");
        for (String word : words) {
            String pigLatinWord = convertToPigLatin(word);
            System.out.print(pigLatinWord + " ");
        }
    }

    public static String convertToPigLatin(String word) {
        char firstChar = Character.toLowerCase(word.charAt(0));
        if (isVowel(firstChar)) {
            return word + "yay";
        } else {
            return word.substring(1) + firstChar + "ay";
        }
    }

    public static boolean isVowel(char ch) {
        ch = Character.toLowerCase(ch);
        return ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u';
    }

    /**
     * Variable Table:
     *
     * | Variable      | Description                                    |
     * |---------------|------------------------------------------------|
     * | `inputString` | The input string entered by the user           |
     * | `words`       | Array of words obtained by splitting the string |
     * | `word`        | The current word being processed               |
     * | `pigLatinWord`| The converted Pig Latin word for the current word |
     * | `firstChar`   | The first character of the current word        |
     */
}
```

- Output

```
Enter a string: Hello There
ellohay heretay
```

Program - 5

A computer salesman gets commission on the following basis:
Sales Commission Rate
Rs. 0 - 20,000 = 3%
Rs. 20,000 - 50,000 = 12%
Rs. 50,001 and more = 31%
After accepting the sales as input, calculate and print his commission amount and rate of commission.

- Source Code

```java
import java.util.Scanner;

public class SalesCommission {
    public static void main(String[] args) {
        double commission = 0.0;
        int rate = 0;
        int sale;

        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the sales amount: ");
        sale = scanner.nextInt();

        if (sale <= 20000) {
            rate = 3;
            commission = sale * 0.3;
        }
        if (sale >= 20000 && sale <= 50000) {
            rate = 12;
            commission = sale * 0.12;
        }
        if (sale >= 50000) {
            rate = 31;
            commission = sale * 0.31;
        }

        System.out.println("Commission amount: " + commission);
        System.out.println("Commission rate: " + rate + "%");
    }
}

/**
 * Variable Table:
 *
 * | Variable   | Description                              |
 * |------------|------------------------------------------|
 * | `commission` | The calculated commission amount         |
 * | `rate`       | The commission rate                      |
 * | `sale`       | The sales amount entered by the user     |
 * | `scanner`    | Used to read input from the user         |
 */
```

- Output

```
Enter the sales amount: 560000
```

```
Commission amount: 173600.0
Commission rate: 31%
```

Program - 9

Write a program in Java to accept a string from the user. Count and display the number of upper case characters, the number of lower case characters and the number of vowels as per the user's choice (Using Switch Case)

- Source Code

```java
import java.util.Scanner;

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

        System.out.print("Enter a string: ");
        String inputString = scanner.nextLine();

        System.out.println("Choose an option:");
        System.out.println("1. Count uppercase characters");
        System.out.println("2. Count lowercase characters");
        System.out.println("3. Count vowels");
        int option = scanner.nextInt();

        switch (option) {
            case 1:
                int uppercaseCount = 0;
                for (int i = 0; i < inputString.length(); i++) {
                    if (Character.isUpperCase(inputString.charAt(i))) {
                        uppercaseCount++;
                    }
                }
                System.out.println("Uppercase count: " + uppercaseCount);
                break;
            case 2:
                int lowercaseCount = 0;
                for (int i = 0; i < inputString.length(); i++) {
                    if (Character.isLowerCase(inputString.charAt(i))) {
                        lowercaseCount++;
                    }
                }
                System.out.println("Lowercase count: " + lowercaseCount);
                break;
            case 3:
                int vowelCount = 0;
                for (int i = 0; i < inputString.length(); i++) {
                    char ch = inputString.charAt(i);
                    if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u' ||
                            ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U') {
                        vowelCount++;
                    }
                }
                System.out.println("Vowel count: " + vowelCount);
                break;
            default:
                System.out.println("Invalid option");
        }
    }
}

/**
 * Variable Table:
```

```
 *
 * | Variable         | Description                          |
 * |------------------|--------------------------------------|
 * | `inputString`    | The input string entered by the user |
 * | `option`         | The selected option                  |
 * | `scanner`        | Used to read input from the user     |
 * | `uppercaseCount` | The count of uppercase characters    |
 * | `lowercaseCount` | The count of lowercase characters    |
 * | `vowelCount`     | The count of vowel characters        |
 */
```

- Output

```
Enter a string: MY name is MaHaRsH
Choose an option:
1. Count uppercase characters
2. Count lowercase characters
3. Count vowels
1
Uppercase count: 6
```

Program - 15

Write a program by using a class in Java with the following specifications:

Class name — StringOP

Data members:

String str
String rev

Member functions:

StringOP() — to initialize variable
void input() — to input a the variables
void process() — to reverse the string str and put it into string rev

- Source Code

```
import java.util.Scanner;

public class StringOP {
    private String str;
    private String rev;

    public StringOP() {
        str = "";
        rev = "";
    }

    public void input() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a sentence: ");
        str = scanner.nextLine();
        scanner.close();
    }

    public void process() {
        StringBuilder sb = new StringBuilder(str);
        rev = sb.reverse().toString();
    }

    public static void main(String[] args) {
        StringOP stringOP = new StringOP();
        stringOP.input();
        stringOP.process();
        System.out.println("Original sentence: " + stringOP.str);
        System.out.println("Reversed sentence: " + stringOP.rev);
    }
}

/**
 * Variable Table:
 *
 * | Variable   | Description                          |
```

```
 * |------------|----------------------------------------|
 * | `str`      | The original input sentence            |
 * | `rev`      | The reversed version of the input sentence |
 * | `scanner`  | Used to read input from the user       |
 * | `stringOP` | An instance of the `StringOP` class    |
 * | `sb`       | A `StringBuilder` to reverse the sentence |
 */
```

- Output

```
Enter a sentence: The great pyramids
Original sentence: The great pyramids
Reversed sentence: sdimaryp taerg ehT
```

Program - 6

Write a program in Java to enter a natural number. display all the possible combinations of consecutive natural number which adds up to give the sum equal to the original number.

- Source Code

```java
import java.util.*;

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

        System.out.print("Enter a positive integer: ");
        int n = sc.nextInt();

        for (int i = 1; i <= n / 2 + 1; i++) {
            int sum = 0;
            int j = i;
            while (sum < n) {
                sum += j;
                j++;
            }
            if (sum == n) {
                for (int k = i; k < j; k++) {
                    System.out.print(k + " ");
                }
                System.out.println();
            }
        }
    }
}

/**
 * Variable Table:
 *
 * | Variable | Description                                |
 * |----------|--------------------------------------------|
 * | `sc`     | Used to read input from the user           |
 * | `n`      | The positive integer entered by the user   |
 * | `i`      | The starting number for the sum of series  |
 * | `j`      | A counter variable for the sum of series   |
 * | `sum`    | The current sum of numbers in the series   |
 * | `k`      | A variable for printing the series numbers |
 */
```

- Output

```
Enter a positive integer: 15
1 2 3 4 5
4 5 6
7 8
```

Program - 3

Write a program in Java to perform all the arithmetic calculations using Switch Case

- Source Code

```java
import java.util.*;

public class switchCalc {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter value of 1st number: ");
        int a = sc.nextInt();

        System.out.println("Enter value of 2nd number: ");
        int b = sc.nextInt();

        System.out.println("Select operation");
        System.out.println("Addition-a: Subtraction-s: Multiplication-m: Division-d: ");
        char ch = sc.next().charAt(0);
        switch(ch) {
            case 'a':
                System.out.println("Sum of the given two numbers: " + (a + b));
                break;

            case 's':
                System.out.println("Difference between the two numbers: " + (a - b));
                break;

            case 'm':
                System.out.println("Product of the two numbers: " + (a * b));
                break;

            case 'd':
                System.out.println("Result of the division: " + (a / b));
                break;

            default:
                System.out.println("Invalid input");
        }
    }
}

/**
 * Variable Table:
 *
 * | Variable | Description                              |
 * |----------|------------------------------------------|
 * | `sc`     | Used to read input from the user         |
 * | `a`      | The value of the first number            |
 * | `b`      | The value of the second number           |
 * | `ch`     | The selected operation character         |
 */
```

- Output

```
Enter value of 1st number:
14
Enter value of 2nd number:
6134
```

```
Select operation
Addition-a: Subtraction-s: Multiplication-m: Division-d:
m
Product of the two numbers: 85876
```

Program - 11

Write a program in Java (Using Scanner Class) to enter a token/words in a mixed case and display the new token after deleting all the vowels. Finally arrange the new token in alphabetical order of letters.

- Source Code

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

public class vowelDeleter {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a token/words in mixed case: ");
        String inputToken = scanner.nextLine();

        String newToken = deleteVowels(inputToken);
        System.out.println("New token after deleting vowels: " + newToken);

        char[] charArray = newToken.toCharArray();
        Arrays.sort(charArray);
        String sortedToken = new String(charArray);
        System.out.println("New token in alphabetical order: " + sortedToken);
    }

    public static String deleteVowels(String token) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < token.length(); i++) {
            char ch = token.charAt(i);
            if (!isVowel(ch)) {
                sb.append(ch);
            }
        }
        return sb.toString();
    }

    public static boolean isVowel(char ch) {
        ch = Character.toLowerCase(ch);
        return ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u';
    }
}

/**
 * Variable Table:
 *
 * | Variable      | Description                          |
 * |---------------|--------------------------------------|
 * | `scanner`     | Used to read input from the user     |
 * | `inputToken`  | The input token/words                |
 * | `newToken`    | The token after deleting vowels      |
 * | `charArray`   | Array representation of `newToken`   |
 * | `sortedToken` | `newToken` sorted in alphabetical order|
 */
```

- Output

```
Enter a token/words in mixed case: The wall is made out of cememt
New token after deleting vowels: Th wll s md t f cmmt
New token in alphabetical order:       Tcdfhllmmmsttw
```