APEX CODING

## // --- 1. Mathematical Scientific Calculator

```
public class Calculator {
    public Double firstnumber { get; set; }
    public Double secondnumber { get; set; }
    public Double result { get; set; }
    public String operationName { get; set; }

    public List<String> history { get; set; }

    public Calculator() {
        history = new List<String>();
    }

    private void log(String opName, String message) {
        operationName = opName;
        if (result != null)
            history.add(message + ' = ' + result);
        else
            history.add(message + ' = Invalid');
    }

    public void Addition() {
        result = firstnumber + secondnumber;

        log('Addition', 'Addition of ' + firstnumber + ' and ' + secondnumber);
    }

    public void Subtraction() {
        result = firstnumber - secondnumber;
        log('Subtraction', 'Subtraction of ' + firstnumber + ' and ' + secondnumber);
    }

    public void Multiplication() {
        result = firstnumber * secondnumber;
        log('Multiplication', 'Multiplication of ' + firstnumber + ' and ' + secondnumber);
    }

    public void Division() {
        result = (secondnumber != 0) ? firstnumber / secondnumber : null;
        log('Division', 'Division of ' + firstnumber + ' by ' + secondnumber);
    }

    public void Modulus() {
        result = Math.mod(firstnumber.intValue(), secondnumber.intValue());
        log('Modulus', 'Modulus of ' + firstnumber + ' % ' + secondnumber);
```

```
}

public void power() {
    result = Math.pow(firstnumber, secondnumber);
    log('Power', firstnumber + ' ^ ' + secondnumber);
}

public void square_root1() {
    result = Math.sqrt(firstnumber);
    log('Square Root', 'Square root of ' + firstnumber);
}

public void square_root2() {
    result = Math.sqrt(secondnumber);
    log('Square Root', 'Square root of ' + secondnumber);
}

public void square1() {
    result = firstnumber * firstnumber;
    log('Square', 'Square of ' + firstnumber);
}

public void square2() {
    result = secondnumber * secondnumber;
    log('Square', 'Square of ' + secondnumber);
}

public void sine1() {
    result = Math.sin(firstnumber * Math.PI / 180);
    log('Sine', 'Sine of ' + firstnumber + '°');
}

public void sine2() {
    result = Math.sin(secondnumber * Math.PI / 180);
    log('Sine', 'Sine of ' + secondnumber + '°');
}

public void cosine1() {
    result = Math.cos(firstnumber * Math.PI / 180);
    log('Cosine', 'Cosine of ' + firstnumber + '°');
}

public void cosine2() {
    result = Math.cos(secondnumber * Math.PI / 180);
    log('Cosine', 'Cosine of ' + secondnumber + '°');
}

public void tangent1() {
    result = Math.tan(firstnumber * Math.PI / 180);
    log('Tangent', 'Tangent of ' + firstnumber + '°');
}

public void tangent2() {
    result = Math.tan(secondnumber * Math.PI / 180);
    log('Tangent', 'Tangent of ' + secondnumber + '°');
}

public void log1() {
    result = (firstnumber > 0) ? Math.log(firstnumber) / Math.log(10) : null;
    log('Log', 'Log base 10 of ' + firstnumber);
}

public void log2() {
    result = (secondnumber > 0) ? Math.log(secondnumber) / Math.log(10) : null;
    log('Log', 'Log base 10 of ' + secondnumber);
}

public void ln1() {
    result = (firstnumber > 0) ? Math.log(firstnumber) : null;
    log('Ln', 'Ln of ' + firstnumber);
}

public void ln2() {
    result = (secondnumber > 0) ? Math.log(secondnumber) : null;
    log('Ln', 'Ln of ' + secondnumber);
}

public void exp1() {
    result = Math.exp(firstnumber);
    log('Exponential', 'e^' + firstnumber);
}

public void exp2() {
    result = Math.exp(secondnumber);
    log('Exponential', 'e^' + secondnumber);
}

public void abs1() {
    result = Math.abs(firstnumber);
    log('Absolute', 'Absolute of ' + firstnumber);
}

public void abs2() {
    result = Math.abs(secondnumber);
    log('Absolute', 'Absolute of ' + secondnumber);
}
public void reset() {
    firstnumber = null;
    secondnumber = null;
    result = null;
    operationName = null;
    history.clear();
}
}
```

## // --- 2. Generate Student Mark Sheet

```
public class Marksheet {
    public void generate(String name, Integer[] marks) {
        Integer total = 0;
        for(Integer mark : marks) total += mark;
        Double percentage = total / (Double)marks.size();
        System.debug('Student: ' + name);
        System.debug('Total Marks: ' + total);
        System.debug('Percentage: ' + percentage + '%');
    }
}

// To execute in Anonymous Window:
Marksheet ms = new Marksheet();
ms.generate('John Doe', new Integer[]{85, 90, 78, 92});
```

## // --- 3. Greatest Among Three Numbers

```
public class Maximum {
    public void compare(Integer x, Integer y, Integer z) {
        if (x == null || y == null || z == null) {
            System.debug('Error: One or more input values are null.');
            return;
        }
        if (x == y && y == z) {
            System.debug(x + ' is the greatest number (All numbers are equal)');
        } else if (x > y && x > z) {
            System.debug(x + ' is the greatest number');
        } else if (y > x && y > z) {
            System.debug(y + ' is the greatest number');
        } else {
            System.debug(z + ' is the greatest number');
        }
    }
}
```

```
// To execute in Anonymous Window:
Maximum m = new Maximum();
m.compare(5, 9, 3); // Example to test
```

## // --- 4. Electricity Bill Calculator

```
public class ElectricityBill {
    public void calculateBill(Integer units) {
        if (units == null || units < 0) {
            System.debug('Error: Invalid input. Units cannot be null or negative.');
            return;
        }
        Double rate = 0;
        if (units <= 100) {
            rate = 5.0;
        } else if (units <= 200) {
            rate = 7.5;
        } else {
            rate = 10.0;
        }
        Double bill = units * rate;

        System.debug('Units Consumed: ' + units);
        System.debug('Rate per Unit: ₹' + rate);
        System.debug('Total Bill: ₹' + bill);
    }
}

// To execute in Anonymous Window:
ElectricityBill eb = new ElectricityBill();
eb.calculateBill(150); // Example of 150 units consumed
```

## // --- 5. Celsius to Fahrenheit Conversion

```
public class TemperatureConverter {
    public Double temperature { get; set; }
    public String scale { get; set; }
    public String result { get; set; }

    public void convert(){
        if(scale == 'C'){
            Double farenheit = 32 + (1.8*temperature);
            result = temperature + ' C = '+farenheit+' F';
        }
        else if(scale == 'F'){
            Double celsius = (temperature - 32)/1.8;
            result = temperature + ' F = '+celsius+' C';
        }
        else{
            result = 'Invalid scale.';
        }
    }
}

<apex:page controller="TemperatureConverter">
    <h1>Temperature Converter</h1>
    <apex:form >
            <apex:pageBlock title="Enter Temperature">
            <apex:pageBlockSection columns="1" >
                        <apex:inputText label="Temperature" value="{!temperature}" />
                <apex:selectList value="{!scale}" size="1" label="Scale" >
                    <apex:selectOption itemLabel="Celsius" itemValue="C" />
                    <apex:selectOption itemLabel="Farenheit" itemValue="F" />
                </apex:selectList>
                <apex:commandButton value="Convert" action="{!convert}"/>
            </apex:pageBlockSection>
            <apex:outputPanel >
                        <apex:outputText value="{!result}" />
            </apex:outputPanel>
        </apex:pageBlock>
    </apex:form>
</apex:page>
```

## // --- 6. Currency Converter

```
public class CurrencyConverter {

    public void convertCurrency(String fromCurrency, String toCurrency, Double amount) {
        Double conversionRate = 0.0;

        if (fromCurrency == 'INR' && toCurrency == 'USD') {
            conversionRate = 0.012;
        } else if (fromCurrency == 'USD' && toCurrency == 'INR') {
            conversionRate = 83.38;
        } else if (fromCurrency == 'INR' && toCurrency == 'Pound') {
            conversionRate = 0.0096;
        } else if (fromCurrency == 'Pound' && toCurrency == 'INR') {
            conversionRate = 104.63;
        } else if (fromCurrency == 'USD' && toCurrency == 'Pound') {
            conversionRate = 0.80;
        } else if (fromCurrency == 'Pound' && toCurrency == 'USD') {
            conversionRate = 1.25;
        } else if (fromCurrency == toCurrency) {
            System.debug('The currencies are the same: ' + fromCurrency);
            return;
        } else {
            System.debug('Conversion rate not available for ' + fromCurrency + ' to ' +
toCurrency);
            return;
        }

        Double convertedAmount = amount * conversionRate;
        System.debug(amount + ' ' + fromCurrency + ' is equal to ' + convertedAmount + ' ' +
toCurrency);
    }
}

// To execute in Anonymous Window:
// Convert 1000 INR to USD
CurrencyConverter cc = new CurrencyConverter();
cc.convertCurrency('INR', 'USD', 1000.0);

// Convert 500 USD to INR
cc.convertCurrency('USD', 'INR', 500.0);
```

## // --- 7. Compound Interest

```
public class CompoundInterest {
    public void calculate(Double p, Double r, Integer t) {
        // Formula for compound interest: A = P(1 + r/n)^(nt)
        Double amount = p * Math.pow((1 + r / 100), t); // Compound interest formula
        Double ci = amount - p; // Compound interest is the total amount minus the principal

        System.debug('Principal: ₹' + p);
        System.debug('Annual Interest Rate: ' + r + '%');
        System.debug('Time Period: ' + t + ' years');
        System.debug('Compound Interest: ₹' + ci);
        System.debug('Total Amount (Principal + Interest): ₹' + amount);
    }
}

// To execute in Anonymous Window:
// Example: Calculate Compound Interest for a principal of ₹10000 at 5% for 2 years
CompoundInterest ci = new CompoundInterest();
ci.calculate(10000.0, 5.0, 2);
```

## // --- 8. Area of Geometry

```
public class GeometryArea {
    public void calculateCircleArea(Double radius) {
        Double area = Math.PI * Math.pow(radius, 2);
        System.debug('Area of Circle: ' + area);
    }

    public void calculateRectangleArea(Double length, Double width) {
        Double area = length * width;
        System.debug('Area of Rectangle: ' + area);
    }

    public void calculateTriangleArea(Double base, Double height) {
        Double area = 0.5 * base * height;
        System.debug('Area of Triangle: ' + area);
    }

    public void calculateSquareArea(Double side) {
        Double area = Math.pow(side, 2);
        System.debug('Area of Square: ' + area);
    }
}

// To execute in Anonymous Window:
GeometryArea ga = new GeometryArea();
ga.calculateCircleArea(5.0);       // Circle with radius 5
ga.calculateRectangleArea(6.0, 4.0); // Rectangle with length 6 and width 4
ga.calculateTriangleArea(6.0, 3.0);  // Triangle with base 6 and height 3
ga.calculateSquareArea(4.0);       // Square with side 4
```

## // --- 9. BFS Traversal

```
public class BFS {
    public void bfs(List<List<Integer>> graph, Integer start) {
        Set<Integer> visited = new Set<Integer>();
        Queue<In           teger> queue = new Queue<Integer>();
        queue.add(start);
        visited.add(start);

        while(!queue.isEmpty()) {
            Integer node = queue.poll();
            System.debug('Visited node: ' + node);

            // Visit all neighbors of the current node
            for(Integer neighbor : graph.get(node)) {
                if(!visited.contains(neighbor)) {
                    queue.add(neighbor);
                    visited.add(neighbor);
                }
            }
        }
    }
}

// To execute in Anonymous Window:
BFS bfs = new BFS();
bfs.bfs(new List<List<Integer>>{
    new List<Integer>{1, 2},  // Node 0's neighbors
    new List<Integer>{0, 3},  // Node 1's neighbors
    new List<Integer>{0, 3},  // Node 2's neighbors
    new List<Integer>{1, 2}   // Node 3's neighbors
}, 0);
```

## // --- 10. Array Element Addition

```
public class ArraySum {
    public void sumArray(Integer[] arr) {
        Integer sum = 0;

        for (Integer i = 0; i < arr.length; i++) {
            sum += arr[i];
        }

        System.debug('Sum of all elements: ' + sum);
    }
}

// To execute in Anonymous Window:
ArraySum as = new ArraySum();
as.sumArray(new Integer[]{1, 2, 3, 4, 5});
```

## // --- 11. Matrix Addition

```
public class MatrixAddition {
    public void addMatrices(List<List<Integer>> mat1, List<List<Integer>> mat2) {
        if (mat1.size() != mat2.size() || mat1[0].size() != mat2[0].size()) {
            System.debug('Matrices must be of the same size');
            return;
        }

        List<List<Integer>> result = new List<List<Integer>>();

        for (Integer i = 0; i < mat1.size(); i++) {
            List<Integer> row = new List<Integer>();
            for (Integer j = 0; j < mat1[i].size(); j++) {
                row.add(mat1[i][j] + mat2[i][j]);
            }
            result.add(row);
        }
        System.debug('Resulting Matrix: ' + result);
    }
}

// To execute in Anonymous Window:
MatrixAddition ma = new MatrixAddition();
List<List<Integer>> mat1 = new List<List<Integer>>{
    new List<Integer>{1, 2},
    new List<Integer>{3, 4}
};
List<List<Integer>> mat2 = new List<List<Integer>>{
    new List<Integer>{5, 6},
    new List<Integer>{7, 8}
};
ma.addMatrices(mat1, mat2);
```

## // --- 12. Matrix Multiplication

```
public class MatrixMultiplication {
    public void multiplyMatrices(List<List<Integer>> mat1, List<List<Integer>> mat2) {
        // Check if the number of columns in mat1 equals the number of rows in mat2
        if (mat1[0].size() != mat2.size()) {
            System.debug('Number of columns in mat1 must be equal to number of rows in
mat2');
            return;
        }

        List<List<Integer>> result = new List<List<Integer>>();
        for (Integer i = 0; i < mat1.size(); i++) {
            List<Integer> row = new List<Integer>();
            for (Integer j = 0; j < mat2[0].size(); j++) {
                row.add(0); // Initialize the row with zeros
            }
            result.add(row);
        }

        for (Integer i = 0; i < mat1.size(); i++) {
            for (Integer j = 0; j < mat2[0].size(); j++) {
                for (Integer k = 0; k < mat2.size(); k++) {
                    result[i][j] += mat1[i][k] * mat2[k][j];
                }
            }
        }
        System.debug('Resulting Matrix: ' + result);
    }
}

// To execute in Anonymous Window:
MatrixMultiplication mm = new MatrixMultiplication();
List<List<Integer>> mat1 = new List<List<Integer>>{
    new List<Integer>{1, 2},
    new List<Integer>{3, 4}
};
List<List<Integer>> mat2 = new List<List<Integer>>{
    new List<Integer>{5, 6},
    new List<Integer>{7, 8}
};
mm.multiplyMatrices(mat1, mat2);
```

## // --- 13. First 100 Natural Numbers

```
public class NaturalNumbers {
    public void printNaturalNumbers() {
        for (Integer i = 1; i <= 100; i++) {
            System.debug(i);         }
    }
}

// To execute in Anonymous Window:
NaturalNumbers nn = new NaturalNumbers();
nn.printNaturalNumbers();
```

## // --- 14. Fibonacci Series

```
public class Fibonacci {
    public void generate(Integer n) {
        if (n <= 0) {
            System.debug('Please provide a positive number.');
            return;
        }
        Integer a = 0, b = 1;
        if (n >= 1) System.debug(a);
        if (n >= 2) System.debug(b);

        // Generate Fibonacci numbers from the 3rd number onward
        for (Integer i = 2; i < n; i++) {
            Integer next = a + b;
            System.debug(next);
            a = b;
            b = next;
        }
    }
}

// To execute in Anonymous Window:
Fibonacci fib = new Fibonacci();
fib.generate(10);
```

## // --- 15. Prime Check

```
public class PrimeCheck {
    public void checkPrime(Integer num) {
        Boolean isPrime = true;
        if (num <= 1) {
            isPrime = false; // 0 and 1 are not prime
        }
        for (Integer i = 2; i <= Math.sqrt(num); i++) {
            if (num % i == 0) { // check if divisible
                isPrime = false;
                break;
            }
        }
        System.debug('Is ' + num + ' prime? ' + isPrime);
    }
}

// To execute in Anonymous Window:
PrimeCheck pc = new PrimeCheck();
pc.checkPrime(7);
```