**Q1)** **create an application to calculate compound intrest on salesforce.com using apex programming language**

**Application Overview**

**🎯 Features:**

* Inputs: **Principal**, **Rate of Interest**, **Time (Years)**, **Number of Times Interest is Compounded per Year**
* Formula:

A=P×(1+rn)ntA = P \times \left(1 + \frac{r}{n} \right)^{nt}A=P×(1+nr​)nt Compound Interest=A−P\text{Compound Interest} = A - PCompound Interest=A−P

**🔹 Step 1: Apex Controller – CompoundInterestCont**

public class CompoundInterestCont {

public Decimal principal { get; set; }

public Decimal rate { get; set; }

public Decimal duration { get; set; } // Renamed from 'time'

public Integer n { get; set; }

public Decimal compoundInterest { get; set; }

public Decimal amount { get; set; }

public void calculate() {

if(principal > 0 && rate > 0 && duration > 0 && n > 0) {

Double P = (Double) principal;

Double r = (Double) (rate / 100);

Double t = (Double) duration;

Double nt = n \* t;

Double baseVal = 1 + (r / n);

Double A = P \* Math.pow(baseVal, nt);

amount = (Decimal) A;

compoundInterest = amount - principal;

} else {

amount = 0;

compoundInterest = 0;

}

}

}

✅ FINAL FIXED VISUALFORCE PAGE

xml

Copy

Edit

<apex:page controller="CompoundInterestCont">

<h2>Compound Interest Calculator</h2>

<apex:form>

<apex:pageBlock title="Enter Values">

<apex:pageBlockSection>

<apex:inputText value="{!principal}" label="Principal (P)"/>

<apex:inputText value="{!rate}" label="Rate (%) (r)"/>

<apex:inputText value="{!duration}" label="Time (Years) (t)"/>

<apex:inputText value="{!n}" label="Compounded Times/Year (n)"/>

</apex:pageBlockSection>

<apex:pageBlockSection>

<apex:commandButton value="Calculate" action="{!calculate}"/>

</apex:pageBlockSection>

<apex:pageBlockSection rendered="{!NOT(ISNULL(compoundInterest))}">

<apex:outputText value="Total Amount (A): ₹{!amount}"/><br/>

<apex:outputText value="Compound Interest: ₹{!compoundInterest}"/>

</apex:pageBlockSection>

</apex:pageBlock>

</apex:form>

</apex:page>

Q2) create an application to calculate area of geometry on salesforce.com using apex programming language

**Step 1: Apex Controller – GeometryAreaCont**

apex

CopyEdit

public class GeometryAreaCont {

public String shape { get; set; }

public Decimal length { get; set; }

public Decimal width { get; set; }

public Decimal radius { get; set; }

public Decimal baseVal { get; set; }

public Decimal height { get; set; }

public Decimal area { get; set; }

public List<SelectOption> getShapeOptions() {

return new List<SelectOption>{

new SelectOption('Square', 'Square'),

new SelectOption('Rectangle', 'Rectangle'),

new SelectOption('Circle', 'Circle'),

new SelectOption('Triangle', 'Triangle')

};

}

public void calculate() {

if(shape == 'Square' && length != null) {

area = length \* length;

} else if(shape == 'Rectangle' && length != null && width != null) {

area = length \* width;

} else if(shape == 'Circle' && radius != null) {

area = Math.PI \* Math.pow((Double)radius, 2);

} else if(shape == 'Triangle' && baseVal != null && height != null) {

area = 0.5 \* baseVal \* height;

} else {

area = null;

}

}

}

**🔹 Step 2: Visualforce Page – GeometryAreaPage**

xml

CopyEdit

<apex:page controller="GeometryAreaCont">

<h2>Geometry Area Calculator</h2>

<apex:form>

<apex:pageBlock title="Select Shape and Enter Dimensions">

<apex:pageBlockSection columns="1">

<apex:selectList value="{!shape}" size="1" label="Select Shape">

<apex:selectOptions value="{!shapeOptions}"/>

</apex:selectList>

<!-- Input fields rendered conditionally -->

<apex:inputText value="{!length}" label="Length" rendered="{!shape='Square' || shape='Rectangle'}"/>

<apex:inputText value="{!width}" label="Width" rendered="{!shape='Rectangle'}"/>

<apex:inputText value="{!radius}" label="Radius" rendered="{!shape='Circle'}"/>

<apex:inputText value="{!baseVal}" label="Base" rendered="{!shape='Triangle'}"/>

<apex:inputText value="{!height}" label="Height" rendered="{!shape='Triangle'}"/>

</apex:pageBlockSection>

<apex:pageBlockSection>

<apex:commandButton value="Calculate Area" action="{!calculate}"/>

</apex:pageBlockSection>

<apex:pageBlockSection rendered="{!NOT(ISNULL(area))}">

<apex:outputText value="Area = {!area} square units"/>

</apex:pageBlockSection>

</apex:pageBlock>

</apex:form>

</apex:page>

Q3) create an application to perform BFS on salesforce.com using apex programming language  
  
public class BFSCont {

public String inputGraphText { get; set; }

public String startNode { get; set; }

public String bfsResult { get; set; }

public void performBFS() {

Map<String, List<String>> graph = new Map<String, List<String>>();

Set<String> visited = new Set<String>();

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

List<String> queue = new List<String>(); // simulate queue using List

// Build the graph from input

if (String.isNotBlank(inputGraphText)) {

String[] lines = inputGraphText.split('\n');

for (String line : lines) {

String[] parts = line.split(':');

if (parts.size() == 2) {

String node = parts[0].trim();

List<String> neighbors = new List<String>();

for (String neighbor : parts[1].split(',')) {

neighbor = neighbor.trim();

if (!String.isBlank(neighbor)) {

neighbors.add(neighbor);

}

}

graph.put(node, neighbors);

}

}

}

// BFS algorithm

if (!String.isBlank(startNode) && graph.containsKey(startNode)) {

queue.add(startNode);

visited.add(startNode);

while (!queue.isEmpty()) {

String current = queue.remove(0); // remove first = dequeue

result.add(current);

List<String> neighbors = graph.get(current);

for (String neighbor : neighbors) {

if (!visited.contains(neighbor)) {

visited.add(neighbor);

queue.add(neighbor); // enqueue

}

}

}

}

bfsResult = result.isEmpty() ? 'Invalid start node or empty graph.' : String.join(result, ' → ');

}

}

**Step 2: Visualforce Page – BFSPage**

xml

CopyEdit

<apex:page controller="BFSCont">

<h2>Breadth-First Search (BFS) Traversal</h2>

<apex:form>

<apex:pageBlock title="Graph Input">

<apex:pageBlockSection>

<apex:outputLabel value="Graph (Adjacency List):"/>

<apex:inputTextarea value="{!inputGraphText}" rows="7" cols="60"/>

<apex:outputLabel value="Start Node:"/>

<apex:inputText value="{!startNode}"/>

</apex:pageBlockSection>

<apex:pageBlockSection>

<apex:commandButton value="Run BFS" action="{!performBFS}"/>

</apex:pageBlockSection>

<apex:pageBlockSection rendered="{!NOT(ISNULL(bfsResult))}">

<apex:outputText value="BFS Traversal: {!bfsResult}"/>

</apex:pageBlockSection>

</apex:pageBlock>

</apex:form>

</apex:page>

**Q4) create an application to perform Array Addition on salesforce.com using apex programming language**

**App Goal:**

* Let the user input **two arrays of numbers** (comma-separated).
* Add the arrays **element-wise**.
* Show the result in the UI.

**🔹 Step 1: Apex Controller – ArrayAdditionCont**

apex

CopyEdit

public class ArrayAdditionCont {

public String array1Text { get; set; }

public String array2Text { get; set; }

public String resultText { get; set; }

public void addArrays() {

if (String.isBlank(array1Text) || String.isBlank(array2Text)) {

resultText = 'Please enter both arrays.';

return;

}

List<String> arr1Str = array1Text.split(',');

List<String> arr2Str = array2Text.split(',');

if (arr1Str.size() != arr2Str.size()) {

resultText = 'Arrays must be of the same length.';

return;

}

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

for (Integer i = 0; i < arr1Str.size(); i++) {

try {

Integer val1 = Integer.valueOf(arr1Str[i].trim());

Integer val2 = Integer.valueOf(arr2Str[i].trim());

resultArray.add(val1 + val2);

} catch (Exception e) {

resultText = 'Invalid input. Ensure both arrays contain only numbers.';

return;

}

}

resultText = 'Result: ' + String.join(resultArray, ', ');

}

}

**🔹 Step 2: Visualforce Page – ArrayAdditionPage**

xml

CopyEdit

<apex:page controller="ArrayAdditionCont">

<h2>Array Addition Application</h2>

<apex:form>

<apex:pageBlock title="Enter Two Arrays (Comma-Separated)">

<apex:pageBlockSection>

<apex:inputText value="{!array1Text}" label="Array 1"/>

<apex:inputText value="{!array2Text}" label="Array 2"/>

</apex:pageBlockSection>

<apex:pageBlockSection>

<apex:commandButton value="Add Arrays" action="{!addArrays}"/>

</apex:pageBlockSection>

<apex:pageBlockSection rendered="{!NOT(ISBLANK(resultText))}">

<apex:outputText value="{!resultText}"/>

</apex:pageBlockSection>

</apex:pageBlock>

</apex:form>

</apex:page>

**Q5)** **create an application to perform Matrix Addition on salesforce.com using apex programming language**

* User enters **two matrices** (comma-separated values and semicolon-separated rows).
* Apex logic parses and adds the matrices element-wise.
* Displays result in the UI.

**🔹 Step 1: Apex Controller – MatrixAdditionCont**

apex

CopyEdit

public class MatrixAdditionCont {

public String matrix1Text { get; set; }

public String matrix2Text { get; set; }

public String resultMatrix { get; set; }

public void addMatrices() {

resultMatrix = '';

try {

List<List<Integer>> m1 = parseMatrix(matrix1Text);

List<List<Integer>> m2 = parseMatrix(matrix2Text);

if (m1.size() != m2.size() || m1[0].size() != m2[0].size()) {

resultMatrix = 'Matrices must be of the same dimensions.';

return;

}

List<String> resultRows = new List<String>();

for (Integer i = 0; i < m1.size(); i++) {

List<String> row = new List<String>();

for (Integer j = 0; j < m1[i].size(); j++) {

row.add(String.valueOf(m1[i][j] + m2[i][j]));

}

resultRows.add(String.join(row, ', '));

}

resultMatrix = String.join(resultRows, '\n');

} catch (Exception e) {

resultMatrix = 'Error parsing input. Ensure correct format (e.g., "1,2;3,4").';

}

}

private List<List<Integer>> parseMatrix(String input) {

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

for (String rowStr : input.split(';')) {

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

for (String val : rowStr.split(',')) {

row.add(Integer.valueOf(val.trim()));

}

matrix.add(row);

}

return matrix;

}

}

**🔹 Step 2: Visualforce Page – MatrixAdditionPage**

xml

CopyEdit

<apex:page controller="MatrixAdditionCont">

<h2>Matrix Addition</h2>

<apex:form>

<apex:pageBlock title="Enter Matrices (Format: 1,2;3,4)">

<apex:pageBlockSection>

<apex:outputLabel value="Matrix 1:"/>

<apex:inputTextarea value="{!matrix1Text}" rows="4" cols="40"/>

<apex:outputLabel value="Matrix 2:"/>

<apex:inputTextarea value="{!matrix2Text}" rows="4" cols="40"/>

</apex:pageBlockSection>

<apex:pageBlockSection>

<apex:commandButton value="Add Matrices" action="{!addMatrices}"/>

</apex:pageBlockSection>

<apex:pageBlockSection rendered="{!NOT(ISBLANK(resultMatrix))}">

<apex:outputLabel value="Result Matrix:"/><br/>

<apex:outputText value="{!resultMatrix}" escape="false"/>

</apex:pageBlockSection>

</apex:pageBlock>

</apex:form>

</apex:page>

**Q6)** **create an application to perform Matrix Multiplication on salesforce.com using apex programming language.  
  
Accept two matrices as input from the user (format: 1,2;3,4).**

**Perform matrix multiplication (A x B).**

**Display the resulting matrix.**

**🔹 Step 1: Apex Controller – MatrixMultiplicationCont**

**apex**

**CopyEdit**

public class MatrixMultiplicationCont {

public String matrix1Text { get; set; }

public String matrix2Text { get; set; }

public String resultMatrix { get; set; }

public void multiplyMatrices() {

resultMatrix = '';

try {

List<List<Integer>> m1 = parseMatrix(matrix1Text);

List<List<Integer>> m2 = parseMatrix(matrix2Text);

Integer m1Rows = m1.size();

Integer m1Cols = m1[0].size();

Integer m2Rows = m2.size();

Integer m2Cols = m2[0].size();

// Validate matrix dimensions: A(m x n) \* B(n x p) = C(m x p)

if (m1Cols != m2Rows) {

resultMatrix = 'Matrix 1 columns must equal Matrix 2 rows.';

return;

}

List<String> resultLines = new List<String>();

for (Integer i = 0; i < m1Rows; i++) {

List<String> row = new List<String>();

for (Integer j = 0; j < m2Cols; j++) {

Integer sum = 0;

for (Integer k = 0; k < m1Cols; k++) {

sum += m1[i][k] \* m2[k][j];

}

row.add(String.valueOf(sum));

}

resultLines.add(String.join(row, ', '));

}

resultMatrix = String.join(resultLines, '\n');

} catch (Exception e) {

resultMatrix = 'Error: Invalid matrix format. Use comma for columns, semicolon for rows.';

}

}

private List<List<Integer>> parseMatrix(String input) {

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

for (String rowStr : input.split(';')) {

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

for (String val : rowStr.split(',')) {

row.add(Integer.valueOf(val.trim()));

}

matrix.add(row);

}

return matrix;

}

}

🔹 Step 2: Visualforce Page – MatrixMultiplicationPage

xml

CopyEdit

<apex:page controller="MatrixMultiplicationCont">

<h2>Matrix Multiplication</h2>

<apex:form>

<apex:pageBlock title="Enter Matrices (Format: 1,2;3,4)">

<apex:pageBlockSection>

<apex:outputLabel value="Matrix 1:"/>

<apex:inputTextarea value="{!matrix1Text}" rows="4" cols="40"/>

<apex:outputLabel value="Matrix 2:"/>

<apex:inputTextarea value="{!matrix2Text}" rows="4" cols="40"/>

</apex:pageBlockSection>

<apex:pageBlockSection>

<apex:commandButton value="Multiply Matrices" action="{!multiplyMatrices}"/>

</apex:pageBlockSection>

<apex:pageBlockSection rendered="{!NOT(ISBLANK(resultMatrix))}">

<apex:outputLabel value="Result Matrix:"/><br/>

<apex:outputText value="{!resultMatrix}" escape="false"/>

</apex:pageBlockSection>

</apex:pageBlock>

</apex:form>

</apex:page>

**Q6)** **create an application to calculate first 100 naturals numbers on salesforce.com using apex programming language**Show the first 100 natural numbers (1 to 100).

Display their **sum**.

Simple UI built with Visualforce.

**🔹 Step 1: Apex Controller – NaturalNumbersCont**

apex

CopyEdit

public class NaturalNumbersCont {

public String numbers { get; set; }

public Integer totalSum { get; set; }

public void generateNumbers() {

List<String> numList = new List<String>();

Integer sum = 0;

for (Integer i = 1; i <= 100; i++) {

numList.add(String.valueOf(i));

sum += i;

}

numbers = String.join(numList, ', ');

totalSum = sum;

}

}

**🔹 Step 2: Visualforce Page – NaturalNumbersPage**

xml

CopyEdit

<apex:page controller="NaturalNumbersCont">

<h2>First 100 Natural Numbers</h2>

<apex:form>

<apex:pageBlock title="Generate Numbers">

<apex:pageBlockSection>

<apex:commandButton value="Show Numbers" action="{!generateNumbers}"/>

</apex:pageBlockSection>

<apex:pageBlockSection rendered="{!NOT(ISBLANK(numbers))}">

<apex:outputLabel value="Natural Numbers (1 to 100):"/>

<apex:outputText value="{!numbers}" escape="false"/>

<br/><br/>

<apex:outputLabel value="Sum of Numbers:"/>

<apex:outputText value="{!totalSum}"/>

</apex:pageBlockSection>

</apex:pageBlock>

</apex:form>

</apex:page>

**Q7)** **create an application of fibonacci series on salesforce.com using apex programming language**

User inputs how many Fibonacci numbers they want.

The Apex controller calculates the Fibonacci series.

The result is displayed in the Visualforce page.

🔹 Step 1: Apex Controller – FibonacciCont

apex

CopyEdit

public class FibonacciCont {

public Integer count { get; set; }

public String resultSeries { get; set; }

public void generateFibonacci() {

resultSeries = '';

if (count == null || count <= 0) {

resultSeries = 'Please enter a positive number.';

return;

}

List<String> series = new List<String>();

Integer a = 0;

Integer b = 1;

for (Integer i = 0; i < count; i++) {

series.add(String.valueOf(a));

Integer temp = a + b;

a = b;

b = temp;

}

resultSeries = String.join(series, ', ');

}

}

🔹 Step 2: Visualforce Page – FibonacciPage

xml

CopyEdit

<apex:page controller="FibonacciCont">

<h2>Fibonacci Series Generator</h2>

<apex:form>

<apex:pageBlock title="Enter Number of Terms">

<apex:pageBlockSection>

<apex:inputText value="{!count}" label="How many terms?"/>

<apex:commandButton value="Generate Series" action="{!generateFibonacci}"/>

</apex:pageBlockSection>

<apex:pageBlockSection rendered="{!NOT(ISBLANK(resultSeries))}">

<apex:outputLabel value="Fibonacci Series:"/>

<apex:outputText value="{!resultSeries}" escape="false"/>

</apex:pageBlockSection>

</apex:pageBlock>

</apex:form>

</apex:page>

**Q8)create an application to develop scientific calculator on salesforce.com using apex programming language**

**Updated Apex Controller – ScientificCalculatorCont**

apex

CopyEdit

public class ScientificCalculatorCont {

public Decimal num1 { get; set; }

public Decimal num2 { get; set; }

public String operation { get; set; }

public Decimal result { get; set; }

public String errorMessage { get; set; }

public void calculate() {

try {

if (operation == 'Addition') {

result = num1 + num2;

} else if (operation == 'Subtraction') {

result = num1 - num2;

} else if (operation == 'Multiplication') {

result = num1 \* num2;

} else if (operation == 'Division') {

if (num2 == 0) {

errorMessage = 'Error: Division by zero is not allowed.';

return;

}

result = num1 / num2;

} else if (operation == 'SquareRoot') {

if (num1 < 0) {

errorMessage = 'Error: Cannot take the square root of a negative number.';

return;

}

result = Math.sqrt(num1);

} else if (operation == 'Power') {

// Use Double to handle power operation

result = (Decimal) Math.pow((Double) num1, (Double) num2);

} else if (operation == 'Sine') {

result = Math.sin((Double) num1);

} else if (operation == 'Cosine') {

result = Math.cos((Double) num1);

} else if (operation == 'Tangent') {

result = Math.tan((Double) num1);

} else if (operation == 'Log') {

if (num1 <= 0) {

errorMessage = 'Error: Logarithm undefined for non-positive numbers.';

return;

}

result = Math.log((Double) num1);

} else {

errorMessage = 'Invalid operation';

}

} catch (Exception e) {

errorMessage = 'Error: ' + e.getMessage();

}

}

}

🔹 Step 2: Visualforce Page – ScientificCalculatorPage

xml

CopyEdit

<apex:page controller="ScientificCalculatorCont">

<h2>Scientific Calculator</h2>

<apex:form>

<apex:pageBlock title="Enter Values">

<apex:pageBlockSection>

<apex:inputText value="{!num1}" label="Enter First Number" />

<apex:inputText value="{!num2}" label="Enter Second Number (if applicable)" />

<apex:selectList value="{!operation}" size="1">

<apex:selectOption itemLabel="Select Operation" itemValue="" />

<apex:selectOption itemLabel="Addition" itemValue="Addition" />

<apex:selectOption itemLabel="Subtraction" itemValue="Subtraction" />

<apex:selectOption itemLabel="Multiplication" itemValue="Multiplication" />

<apex:selectOption itemLabel="Division" itemValue="Division" />

<apex:selectOption itemLabel="Square Root" itemValue="SquareRoot" />

<apex:selectOption itemLabel="Power" itemValue="Power" />

<apex:selectOption itemLabel="Sine" itemValue="Sine" />

<apex:selectOption itemLabel="Cosine" itemValue="Cosine" />

<apex:selectOption itemLabel="Tangent" itemValue="Tangent" />

<apex:selectOption itemLabel="Logarithm" itemValue="Log" />

</apex:selectList>

</apex:pageBlockSection>

<apex:pageBlockSection>

<apex:commandButton value="Calculate" action="{!calculate}" rerender="resultSection"/>

</apex:pageBlockSection>

<apex:pageBlockSection id="resultSection">

<apex:outputPanel rendered="{!NOT(ISBLANK(errorMessage))}">

<apex:outputText value="{!errorMessage}" style="color: red;" />

</apex:outputPanel>

<apex:outputPanel rendered="{!NOT(ISBLANK(result))}">

<apex:outputLabel value="Result: " />

<apex:outputText value="{!result}" />

</apex:outputPanel>

</apex:pageBlockSection>

</apex:pageBlock>

</apex:form>

</apex:page>

**Q9)** **create an application to determine whether the given number is prime or not on salesforce.com using apex programming language**public class PrimeNumberCont {

public Integer inputNumber { get; set; }

public String result { get; set; }

public void checkPrime() {

if (inputNumber == null || inputNumber <= 1) {

result = inputNumber + ' is NOT a Prime Number.';

return;

}

Boolean isPrime = true;

for (Integer i = 2; i <= Math.floor(Math.sqrt(Decimal.valueOf(inputNumber))); i++) {

if (Math.mod(inputNumber, i) == 0) {

isPrime = false;

break;

}

}

result = isPrime

? inputNumber + ' is a Prime Number.'

: inputNumber + ' is NOT a Prime Number.';

}

}

Full corrected VF Page:

CopyEdit

<apex:page controller="PrimeNumberController">

<apex:form>

<apex:pageBlock title="Prime Number Checker">

<apex:pageBlockSection>

<apex:inputText value="{!inputNumber}" label="Enter a Number"/>

<apex:commandButton value="Check" action="{!checkPrime}" rerender="resultPanel"/>

</apex:pageBlockSection>

<apex:outputPanel id="resultPanel">

<apex:outputText value="{!result}" style="font-weight:bold; color:blue;" rendered="{!NOT(ISNULL(result))}"/>

</apex:outputPanel>

</apex:pageBlock>

</apex:form>

</apex:page>