MANG2092

Individual Coursework (20%)
Basic VBA Program

Student ID: 35597836

A. Full VBA Routine

A1. Workbook Module (ThisWorkbook)

```
Option Explicit

'====== PART I: Workbook Welcome Message ======

Private Sub Workbook_Open()
    MsgBox "Welcome to the Investment Scenario Analysis Tool!" & vbNewLine & _
        "Go to the Dashboard sheet to explore the tool and use the button to start your calculations!", _
        vbInformation, "Hello!"
End Sub
```

A2. UserForm Module (ReturnCalculationForm)

A2.1. Main Calculation Routine

```
Option Explicit

'====== PART II: Main Calculation Routine (Validation, Prcessing, Calculating, Output, Logging, Extension) ======

Private Sub cmdCalculate_Click()

'1. DECLARE VARIABLES

    Const total_fund As Double = 10000000000# '£10 billion total capital

    '---User inputs for class allocation (%)---
    Dim inputClass1 As Double, inputClass2 As Double, inputClass3 As Double
    Dim years As Long

    '---Converted to £ amounts of each class based on the percentages
    Dim amtClass1 As Double, amtClass2 As Double, amtClass3 As Double
```

```
    '---Return Rates (converted to decimal form)---
    Dim rAI As Double, rBanks As Double, rEnergy As Double
    Dim rSupply As Double, rRealEstate As Double, rAuto As Double

    '---Sub-sector Returns by Investment Class---
    'Class 1:
    Dim retAI1 As Double
    Dim retEnergy1 As Double
    Dim retRealEstate1 As Double
    'Class 2:
    Dim retAI2 As Double
    Dim retBanks2 As Double
    Dim retEnergy2 As Double
    'Class 3:
    Dim retEnergy3 As Double
    Dim retSupply3 As Double
    Dim retAuto3 As Double

    '---Total Return per Investment Class---
    Dim class1Return As Double
    Dim class2Return As Double
    Dim class3Return As Double

    '---Total Portfolio Return---
    Dim totalReturn As Double

'2. BASIC VALIDATION - check if inputs look correct
    '---Scenario selection check (for auto-fill mode)---
    If chkDefault.Value = True Then
        If optWorst.Value = False And optBest.Value = False Then
```

```
        MsgBox "Because Auto-Fill is enabled, please select Worst Case or Best Case before calculating.", vbExclamation, "Scenario Required"
        Exit Sub
      End If
    End If

    '---Check all inputs (class %, years, return rates) are numbers---
    If Not IsNumeric(txtClass1.Value) Or _
       Not IsNumeric(txtClass2.Value) Or _
       Not IsNumeric(txtClass3.Value) Or _
       Not IsNumeric(txtYears.Value) Or _
       Not IsNumeric(txtAI.Value) Or _
       Not IsNumeric(txtBanks.Value) Or _
       Not IsNumeric(txtEnergy.Value) Or _
       Not IsNumeric(txtSupply.Value) Or _
       Not IsNumeric(txtRealEstate.Value) Or _
       Not IsNumeric(txtAuto.Value) Then

    MsgBox "Please enter valid numbers in textboxes." & vbNewLine & _
           "Use digits only (e.g., 10000 or 7.5). No text, symbols, %, commas, or blanks.", _
           vbCritical, "Invalid Input"
      Exit Sub
    End If

'3. CONVERT TEXTBOX VALUES TO NUMBERS

    '---Convert investment amounts and years to numeric values---
    inputClass1 = CDbl(txtClass1.Value)
    inputClass2 = CDbl(txtClass2.Value)
    inputClass3 = CDbl(txtClass3.Value)
    years = CDbl(txtYears.Value)
```

```
    '---Convert return rates (%) to decimals for calculation---
    rAI = CDbl(txtAI.Value) / 100
    rBanks = CDbl(txtBanks.Value) / 100
    rEnergy = CDbl(txtEnergy.Value) / 100
    rSupply = CDbl(txtSupply.Value) / 100
    rRealEstate = CDbl(txtRealEstate.Value) / 100
    rAuto = CDbl(txtAuto.Value) / 100

'4. EXTRA VALIDATION - negative values, limits

    '---No negative class percentages allowed---
    If inputClass1 < 0 Or inputClass2 < 0 Or inputClass3 < 0 Then
        MsgBox "Class allocation percentages cannot be negative.", vbCritical
        Exit Sub
    End If

    '---No negative return rates allowed---
    If txtAI.Value < 0 Or txtBanks.Value < 0 Or txtEnergy.Value < 0 Or _
       txtSupply.Value < 0 Or txtRealEstate.Value < 0 Or txtAuto.Value < 0 Then
        MsgBox "Return rates cannot be negative numbers.", vbCritical
        Exit Sub
    End If

    '---Years must be positive---
    If years <= 0 Then
        MsgBox "Please enter a positive number of years.", vbCritical
        Exit Sub
    End If

    '---Total % allocation cannot exceed 100%---
    If inputClass1 + inputClass2 + inputClass3 > 100 Then
        MsgBox "Total allocation cannot exceed 100% of available funds.", vbCritical
```

```
        Exit Sub
    End If

'5. CONVERT % CLASS ALLOCATIONS INTO £ AMOUNTS
    amtClass1 = total_fund * (inputClass1 / 100)
    amtClass2 = total_fund * (inputClass2 / 100)
    amtClass3 = total_fund * (inputClass3 / 100)



'6. COMPOUND INTEREST CALCULATION (RETURNS)

' Using standard compound interest model: FV = PV × (1 + r)^t
' where:
' FV = future value
' PV: present value
' r: annual return rate in decimal
' t: number of years

'-> The return (profit): Return = FV - PV = PV × (1 + r)^t - PV

    '---Class 1 Returns---

    'AI = 20%
    retAI1 = (amtClass1 * 0.2) * ((1 + rAI) ^ years) - (amtClass1 * 0.2)

    'Energy = 50%
    retEnergy1 = (amtClass1 * 0.5) * ((1 + rEnergy) ^ years) - (amtClass1 * 0.5)

    'Real Estate = 30%
    retRealEstate1 = (amtClass1 * 0.3) * ((1 + rRealEstate) ^ years) - (amtClass1 * 0.3)

    '---Class 2 Returns---
```

```
    'AI = 20%
    retAI2 = (amtClass2 * 0.2) * ((1 + rAI) ^ years) - (amtClass2 * 0.2)

    'Banks = 60%
    retBanks2 = (amtClass2 * 0.6) * ((1 + rBanks) ^ years) - (amtClass2 * 0.6)

    'Energy = 20%
    retEnergy2 = (amtClass2 * 0.2) * ((1 + rEnergy) ^ years) - (amtClass2 * 0.2)

    '---Class 3 Returns---

    'Energy = 25%
    retEnergy3 = (amtClass3 * 0.25) * ((1 + rEnergy) ^ years) - (amtClass3 * 0.25)

    'Global Supply Chain = 25%
    retSupply3 = (amtClass3 * 0.25) * ((1 + rSupply) ^ years) - (amtClass3 * 0.25)

    'Automotive = 50%
    retAuto3 = (amtClass3 * 0.5) * ((1 + rAuto) ^ years) - (amtClass3 * 0.5)

'7. TOTAL RETURN

    '---Total Return per Class---

    class1Return = retAI1 + retEnergy1 + retRealEstate1
    class2Return = retAI2 + retBanks2 + retEnergy2
    class3Return = retEnergy3 + retSupply3 + retAuto3

    '---Total Portfolio Return---

    totalReturn = class1Return + class2Return + class3Return
```

```
'8. LOG RESULTS IN WORKSHEET - "Investment_Log" worksheet

    Dim ws As Worksheet
    Dim nextrow As Long

    'Set worksheet object
    Set ws = ThisWorkbook.Worksheets("Investment_Log")

    'Find next empty row in Column D
    'Goes to bottom of column D, moves up to last used cell, then adds 1 -> value of nextrow
    nextrow = ws.Cells(ws.Rows.Count, "D").End(xlUp).Row + 1

    'Write entire row of data in ONE line (14 columns)
    'Write all inputs and outputs into the log in a single row
    ws.Range("A" & nextrow & ":P" & nextrow).Value = Array( _
        Now, Environ$("Username"), years, inputClass1, inputClass2, inputClass3, _
        txtAI.Value, txtBanks.Value, txtEnergy.Value, txtSupply.Value, txtRealEstate.Value,
txtAuto.Value, _
        class1Return, class2Return, class3Return, totalReturn _
)

    'Display the number as pounds (£), with commas, and 2 decimals
    ws.Cells(nextrow, 13).NumberFormat = "£#,##0.00"
    ws.Cells(nextrow, 14).NumberFormat = "£#,##0.00"
    ws.Cells(nextrow, 15).NumberFormat = "£#,##0.00"
    ws.Cells(nextrow, 16).NumberFormat = "£#,##0.00"

    'Display the date & time in this format: day/month/year hour:minutes
    ws.Cells(nextrow, 1).NumberFormat = "dd/mm/yyyy hh:mm"

'9. DISPLAY RESULTS TO USER IN MSGBOX
```

```
    MsgBox _
    "Investment Results (" & years & " years)" & vbNewLine & vbNewLine & _
    "Class 1 Return:   £" & Format(class1Return, "#,##0.00") & vbNewLine & _
    "Class 2 Return:   £" & Format(class2Return, "#,##0.00") & vbNewLine & _
    "Class 3 Return:   £" & Format(class3Return, "#,##0.00") & vbNewLine & _
    "Total Return: £" & Format(totalReturn, "#,##0.00"), _
    vbInformation, "Scenario Results"

'10. Allow Multiple Calculations

 Dim repeatCalculation As VbMsgBoxResult
 repeatCalculation = MsgBox("Would you like to calculate another scenario?", vbYesNo + vbQuestion, "Try Again")

 If repeatCalculation = vbYes Then 'User does want another calculation
    'Reset: clear all input boxes then the user can input again
    txtClass1.Value = ""
    txtClass2.Value = ""
    txtClass3.Value = ""
    txtYears.Value = ""
    txtAI.Value = ""
    txtBanks.Value = ""
    txtEnergy.Value = ""
    txtSupply.Value = ""
    txtRealEstate.Value = ""
    txtAuto.Value = ""
    optWorst.Value = False
    optBest.Value = False
    chkDefault.Value = False
    FrameScenario.Enabled = False
```

```
    Exit Sub 'Return control to the UserForm

Else
    'User does not want another calculation, then close the UserForm
    Unload Me
End If

End Sub
```

A2.2. Auto-Fill Scenario Controls

```
'====== PART III: Managing Worst/Best Case Auto-Fill ======

Private Sub chkDefault_Click()
    'Enable/disable scenario options depending on whether auto-fill mode is used
    FrameScenario.Enabled = chkDefault.Value

    'If auto-fill is turned off, clear the scenario choices and reset all rate fields to blanks
    If chkDefault.Value = False Then
        optWorst.Value = False
        optBest.Value = False

        txtAI.Value = ""
        txtBanks.Value = ""
        txtEnergy.Value = ""
        txtSupply.Value = ""
        txtRealEstate.Value = ""
        txtAuto.Value = ""
    End If
End Sub

Private Sub optWorst_Click()
    'If auto-fill is active, load the minimum return rates(%)(given by SW's industry analytics)
```

```
    If chkDefault.Value = True Then
        txtAI.Value = 3
        txtBanks.Value = 2
        txtEnergy.Value = 0
        txtSupply.Value = 6
        txtRealEstate.Value = 5
        txtAuto.Value = 10
    End If
End Sub

Private Sub optBest_Click()
    'If auto-fill is active, load the maximum return rates(%)(given by SW's industry analytics)
    If chkDefault.Value = True Then
        txtAI.Value = 8
        txtBanks.Value = 5
        txtEnergy.Value = 8
        txtSupply.Value = 10
        txtRealEstate.Value = 9
        txtAuto.Value = 15
    End If
End Sub
```

A3. Standard Module (ButtonClick)

(for the macro that opens the UserForm)

```
Option Explicit

'====== PART IV: Macro to Open the UserForm for Return Calculation ======

Sub OpenScenarioCalculator()

    ReturnCalculatorForm.Show
```

```
End Sub
```

A4. Standard Module (PublicFunction_CalcReturn)

(for the public function)

```
Option Explicit

'====== PART V: Public Function ======

'EXTENSION: Public Function
'Allows calculation of return from any worksheet or any VBA procedure in the project.
'This lets users call: = CalculateReturn(amount, rate, years) directly from Excel.

Public Function CalculateReturn(amount As Double, rate As Double, years As Long) As Double

    'amount = the money invested in a single sub-sector (entered in pounds, e.g., 2500000)
    'rate   = the annual return rate entered as a percentage (e.g., 5 for 5%)
    'years  = the investment duration in positive years (e.g., 5 or 2.5)

    'The function converts the rate from % to decimal using: rate / 100
    'Then applies STANDARD compound interest to calculate the return profit:
    'The return (profit): Return = FV - PV = PV × (1 + r)^t - PV

    '---Basic Validation---
    If amount < 0 Then
        CalculateReturn = CVErr(xlErrValue)
        Exit Function
    End If

    If rate < 0 Then
        CalculateReturn = CVErr(xlErrValue)
        Exit Function
```

```
    End If

    If years <= 0 Then
        CalculateReturn = CVErr(xlErrValue)
        Exit Function
    End If

    'Convert percentage rate to decimal
    Dim r As Double
    r = rate / 100

    'Calculate return (profit only)
    CalculateReturn = amount * ((1 + r) ^ years) - amount

End Function
```

B. Storyline: Purpose and User Inputs

The purpose of this program is to support Terry in analysing long-term investment outcomes for SW Asset Management's £10 billion fund. The storyline focuses on helping him test different scenarios quickly and clearly, so he can see how the portfolio might perform under different economic conditions when the return rate of each sub-sector changes. To do this, Terry can allocate the capital across three investment classes, each containing specific sub-sectors, and apply different return rates depending on what scenario he wants to test.

The user inputs including Class Allocation (%), Sub-sector Return Rates (%), and Investment Duration (years), reflect the actual decisions Terry needs to make:

- **Class Allocation (%)**: Terry decides how much capital goes into each class because each class behaves differently. Some are more stable, while others can change more with the economy. So the way he allocates the capital affects how safe the portfolio is and how much return it can generate.
- **Sub-sector Return Rates (%)**: These rates change depending on the economic outlook, so Terry may want to test different assumptions to see how sensitive each class is to market conditions.
- **Investment Duration (years)**: Time directly affects compound returns, and Terry may want to compare short-term vs long-term performance.

The Auto-Fill option (Worst Case / Best Case) increases flexibility and convenience. Terry can either enter the rates manually or quickly use the return rates already provided by SW's analytics to test two predefined scenarios.

Once the inputs are entered, the program calculates the return for each class and the total return of all three classes by using the compound interest formula:

Return = PV × (1 + r)^t – PV

(This approach can reflect realistic long-term investment behaviour)

Finally, the results are shown in a message box and logged in the "Investment_Log" worksheet, allowing Terry to track every scenario he tests and compare outcomes more easily.

C. Extension Explanation

C1. Scenario Auto-Fill System (Worst Case / Best Case)

Location: *optWorst_Click, optBest_Click, chkDefault_Click*

I added an Auto-Fill feature in the UserForm which lets Terry apply the minimum or maximum return rates from SW's industry analytics. But this is optional, because he can still type in the rates manually to test other conditions. Overall, this reduces manual typing if he wants to use the given rates, making scenario calculations and comparisons faster and improving the user experience.

C2. Run Another Calculation until the user decides to stop

Location: End of *cmdCalculate_Click* (Part 10)

After each calculation, the program asks whether Terry wants to test another scenario. If he selects "Yes", all input fields reset and the UserForm stays open, allowing continuous testing. This increases flexibility and convenience for users.

C3. Public Function for Return Calculation

Location: Module *PublicFunction_CalcReturn*

I created a public function called CalculateReturn allowing users to calculate returns directly from Excel worksheets by calling it as a formula. This is suitable for larger financial applications because it allows user reuse it outside the UserForm and quickly check the return amount of a single sub-sector (for example, "How much can I earn if I invest £2 billion in AI/IT sector for 3 years?").

C4. Logging System ("Investment_Log" Worksheet)

Location: *cmdCalculate_Click* (Part 8)

All scenario inputs and outputs are saved automatically to a worksheet. This extension improves traceability and allows Terry to review and compare scenarios. Additionally, I added timestamp and username columns to make the log more applicable and useful for larger corporations.

C5. UserForm-Based Data Entry Interface

Location: *ReturnCalculationForm*

Instead of using multiple InputBoxes, I decided to design a structured UserForm to improve the realism and convenience of the tool. It groups inputs clearly, validates entries, and

displays helpful error messages This can improve usability, simplifies data entry, enhances the interface, and makes the tool easier to use and update.

C6. Error Detection for Invalid Inputs

Location: *cmdCalculate_Click* (Part 2–4)

I added several validations checks to make sure the program only runs when all inputs are valid. These checks detect problems such as negative numbers, non-numeric values, or when the class allocation is more than 100%. If more than one error happens at the same time, the program cannot show all the error messages together. Instead, it shows one error first. After Terry fixes that error and clicks Calculate again, the next error message will appear, and so on. The UserForm will not calculate anything until every error is fixed. This makes the tool safer to use and stops the program from producing incorrect results.