Excel Assignment - 21

1. Write a VBA code to enter your name in A1 Cell using Input Box and once you enter the name display a message box that says the name has been entered.

→

```
Sub EnterName()
    Dim userName As String

    ' Prompt user for name using Input Box
    userName = InputBox("Enter your name:", "Name Entry")

    ' Check if user entered a name
    If userName <> "" Then
        ' Enter the name in cell A1
        Range("A1").Value = userName

        ' Display a message box
        MsgBox "Name has been entered: " & userName, vbInformation, "Success"
    Else
        ' Display a message if user did not enter a name
        MsgBox "No name entered. Please try again.", vbExclamation, "Warning"
    End If
End Sub
```

2. What are Userforms? Why are they used? How to fill a list box using for loop.

→

UserForms in VBA (Visual Basic for Applications) are dialog boxes or forms that you can create and customize to interact with users. These forms can include various controls like text boxes, labels, buttons, checkboxes, and more. UserForms provide a way to create custom input interfaces for users to input data, make selections, or trigger specific actions within your VBA macro-enabled applications.

Why are UserForms used? UserForms are used for several reasons:

1. **Improved User Interaction:** UserForms enhance the user experience by providing a more user-friendly and structured way to interact with your VBA applications.
2. **Data Input:** They allow users to input data in a structured manner, reducing the chances of errors.
3. **Custom Interfaces:** You can design custom interfaces tailored to your application's needs, making it more intuitive for users.
4. **Validation:** UserForms make it easier to validate user inputs and guide them through the required steps.
5. **Professional Appearance:** They give your VBA applications a more professional appearance compared to using standard message boxes and input boxes.

How to fill a ListBox using a For Loop in VBA: To fill a ListBox using a For Loop in VBA, you'll first need to create a UserForm with a ListBox control on it. Follow these steps:

1. Open the VBA editor by pressing **ALT + F11**.
2. Insert a new UserForm by right-clicking on any of the project nodes in the Project Explorer, choose **Insert**, and then **UserForm**.

3. Add a ListBox control to the UserForm. You can do this from the toolbox (press **CTRL + T** to show it) and drag the ListBox onto the UserForm.
4. With the ListBox selected, go to the properties window (press **F4** if it's not already open) and set its **Name** property to something like "ListBox1."

Now, you can add the following code to fill the ListBox using a For Loop:

```
Private Sub UserForm_Initialize()
    ' Initialize the UserForm

    Dim i As Integer

    ' Clear the ListBox
    Me.ListBox1.Clear

    ' Fill the ListBox using a For Loop
    For i = 1 To 10
        Me.ListBox1.AddItem "Item " & i
    Next i

End Sub
```

This code initializes the UserForm when it is loaded. It clears the ListBox and then adds 10 items to it using a For Loop. You can customize the loop as per your requirements. To test it, close the VBA editor, right-click on the UserForm in the Project Explorer, and choose **Run UserForm**. The UserForm will appear with the ListBox filled with items.

3. What is an array? Write a VBA code to enter students and their marks from the below table.

→

An array is a data structure that stores a collection of elements, all of the same type, under a single variable name. In VBA, you can create arrays to store and manipulate data efficiently. Arrays can be one-dimensional (a single row or column of elements) or multi-dimensional (containing multiple rows and columns).

Let's consider a table with students and their marks:

Student	Marks
John	85
Jane	92
Bob	78
Alice	88

Now, you can create VBA code to store this information in arrays. Below is an example code that creates arrays for student names and marks, and then outputs the data:

```
Sub EnterStudentMarks()
    ' Declare arrays for student names and marks
    Dim studentNames() As String
    Dim studentMarks() As Integer
```

```
    ' Initialize data
    studentNames = Array("John", "Jane", "Bob", "Alice")
    studentMarks = Array(85, 92, 78, 88)

    ' Display the data using a loop
    Dim i As Integer
    For i = LBound(studentNames) To UBound(studentNames)
        Debug.Print "Student: " & studentNames(i) & ", Marks: " & studentMarks(i)
    Next i

End Sub
```

Explanation of the code:

1. **Dim studentNames() As String** and **Dim studentMarks() As Integer** declare arrays for student names and marks, respectively.
2. **studentNames = Array("John", "Jane", "Bob", "Alice")** initializes the array **studentNames** with student names.
3. **studentMarks = Array(85, 92, 78, 88)** initializes the array **studentMarks** with corresponding marks.
4. The **For** loop iterates through the arrays, and **Debug.Print** is used to output the student names and marks to the Immediate Window in the VBA editor.

4. Use the following data to create a pie chart using VBA code. Use Font - 'Times new Roman', Size -14, Bold, Title - Piechart' and you are per to use colours as per your taste.

→

To create a pie chart using VBA code with the specified formatting, you can use the following example. Let's assume you have the following data:

```
Category  | Value
----------
Category1 | 25
Category2 | 35
Category3 | 20
Category4 | 15
```

Now, you can use the following VBA code to create a pie chart with the specified formatting:

```
Sub CreatePieChart()
    Dim ws As Worksheet
    Dim chartObj As ChartObject
    Dim chartDataRange As Range

    ' Set the worksheet
    Set ws = ThisWorkbook.Sheets("Sheet1") ' Change "Sheet1" to your actual sheet name

    ' Define the data range (assuming data starts from A1)
    Set chartDataRange = ws.Range("A1:B5")

    ' Create a new chart on the worksheet
    Set chartObj = ws.ChartObjects.Add(Left:=100, Width:=375, Top:=75, Height:=225)

    ' Set chart data source
```

```
    chartObj.Chart.SetSourceData Source:=chartDataRange

    ' Set chart type to Pie
    chartObj.Chart.ChartType = xlPie

    ' Set chart title and formatting
    With chartObj.Chart
        .HasTitle = True
        .ChartTitle.Text = "Pie Chart"
        .ChartTitle.Font.Name = "Times New Roman"
        .ChartTitle.Font.Size = 14
        .ChartTitle.Font.Bold = True
    End With

    ' Apply color to data points
    With chartObj.Chart.SeriesCollection(1).Points
        .Item(1).Format.Fill.ForeColor.RGB = RGB(255, 0, 0)  ' Red
        .Item(2).Format.Fill.ForeColor.RGB = RGB(0, 255, 0)  ' Green
        .Item(3).Format.Fill.ForeColor.RGB = RGB(0, 0, 255)  ' Blue
        .Item(4).Format.Fill.ForeColor.RGB = RGB(255, 255, 0) ' Yellow
    End With
End Sub
```

5. Write step by step procedure to protect your workbook using a password.

→Protecting a workbook with a password in Microsoft Excel involves the following steps. Please note that the steps might slightly differ based on the version of Excel you are using, but the general process is similar. Here are the steps for Excel 2016 and later versions:

Step 1: Open Your Workbook

Open the Excel workbook that you want to protect with a password.

Step 2: Save Your Workbook

Before protecting your workbook, it's a good practice to save your changes. Go to **File** and click on **Save** or **Save As** if you want to create a new copy.

Step 3: Protect the Workbook Structure:

1. Go to the **Review** tab on the ribbon.
2. Click on the **Protect Workbook** option.
    - In Excel 2016 and later versions, you might see "Protect Workbook" or "Protect Structure and Windows."
3. In the dialog box that appears, check the box next to "Structure."

Step 4: Set a Password:

1. After checking the "Structure" box, you will be prompted to enter a password.
2. Enter a password and click **OK**.
    - You may be asked to confirm the password. Enter the password again and click **OK**.

Step 5: Save Your Workbook Again:

1. After setting the password, save your workbook again to apply the protection.

2. Close and reopen the workbook to test the password protection.

Step 6: Unprotect the Workbook:

If you want to remove the password protection later, follow these steps:

1. Go to the **Review** tab.
2. Click on **Protect Workbook** and enter the password.
3. Uncheck the box next to "Structure" or select the appropriate option.

Remember to save your workbook after removing the protection.