Prateek Agnihotri

Excel Macros

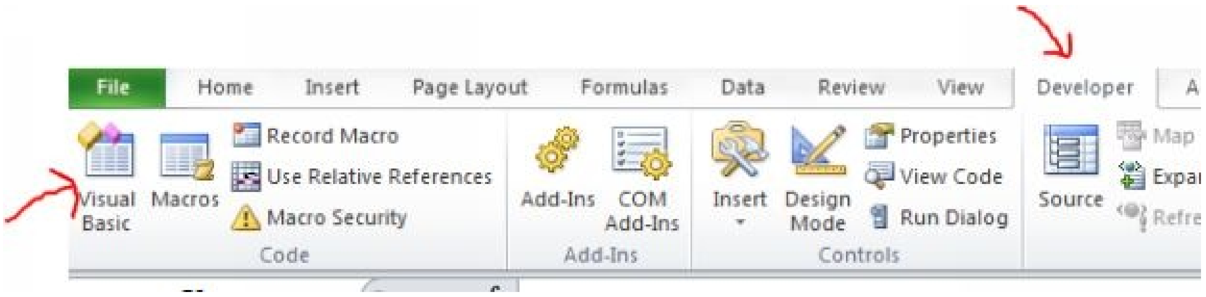
Excel Macros and VBA

VBA AND MACROS

VBA is a major division of the stand-alone Visual Basic programming language. It is integrated into Microsoft Office applications. It is the macro language of Microsoft Office Suite. Previously it was known as xlm.

How to access VBA in MS-Excel-

1. Press ALT+F11.
2. Go To Developer Tab -> Click Visual Basic Icon. See image below.



Every organization in today’s world relies upon various kinds of databases for storage of all kinds of data. Data is the backbone in every aspect of an organization, be it management, marketing, finance, planning, technical and production services, issues, environment etc. Excel can be used as a database or we can call it a spreadsheet application which manipulates the data stored in it or some other database like SQL Server, Oracle database, MySQL etc. Excel has various features like implementing formulas, developing pivots, charts. The most important feature of excel is the macro programming language commonly known as VBA used within excel to develop macros.

VBA means visual basic for applications. Official name is “Visual Basic, Applications Edition. VBA is the vastest language amongst all the high level languages. VBA is an event driven; object oriented programming language from Microsoft that is now primarily used with Microsoft office applications such as MS-Excel, MS-Word and MS-PowerPoint.

Some features of VBA are as follows –

1.) VBA is a high level language which anyone knowing MS applications like excel or word can learn. This language helps in creating a macro which is nothing but a series of instructions for a task to be performed automatically.

2.) VBA enables user to automate repetitive tasks so as to reduce the manual effort.

3.) VBA not only offers macros to be created but also allows user to create UDFs i.e. user defined functions. These functions once built are incorporated in the library with all other excel functions.

4.) VBA works on windows machine so is platform dependent.

5.) VBA helps in eliminating waste and is based on agile methodology.

6.) VBA is an OOP i.e. object oriented language. Everything in VBA is treated as an object.

7.) VBA can be used to connect to any database other than excel itself like MySQL, Oracle etc. It makes the connection with the back end database and manipulates data as required.

8.) VBA can be used with all Microsoft applications like MS-Word, MS-Access, Outlook, MS-Power point etc.

9.) VBA macros are user specific and not author specific. They can be modified, deleted by the user who wants to run it.

10.) VBA in MS-Office provides many inbuilt functions that a user can use to build code in Excel.

11.) VBA allows users to record the macros and then tweak them for specific purposes.

12.) VBA is also used by data analysts, finance & market analysts for data manipulation, modeling etc. Mathematicians use it due to its vast library full of formulas and functions.

13.) VBA allows coder to switch off calculations and sheets update during execution of code which speeds up the processing.

14.) VBA is a Self-interpreted programming language. Compiling is very easy in VBA.

15.) VBA can help built Powerful tools in MS Office using logical programming.

16.) There is a famous one liner about VBA that there is nothing which can’t be done by VBA.

VBA Enables end-user programming and is used in MS Office applications as told above. It encapsulates Formulae and macros for easy tasks. Following are the contents which will be covered in this book –

1.) Concept of Variables and Data Types.

2.) Conditional and Logical operators.

3.) Nested Loops, Switch Cases, conditional statements etc.

4.) Error Handling.

5.) Object handling.

6.) Concept of single and multiple dimensional arrays in VBA.

7.) String manipulation.

8.) Macro Recording.

VBA– Collections

- A Group of Similar Objects that Share Common Properties, Methods and

Events Such as Workbooks, Worksheets, etc. are called Collections.

- Worksheets are a collection of all the Worksheet objects in the active workbook.
- Worksheets (3) refer to the 3rd worksheet of current active workbook.
- Worksheets (“Sheet1”) refer to the worksheet named “Sheet1”.

VBA – Objects

- VBA objects are Worksheet, Workbook, Range, Cell, Chart, Name, etc.
- Worksheets(Sheet1) is an Object Referring to the First Sheet
- Range(“A1:A5”) is an Object Referring to a Range from Row 1 , Column 1 to Row 5, Column 1
- Range(“A1:B5”) is an Object Referring to a Range from Row 1 , Column 1 to Row 5, Column 2.
- Cells (1,1) or Range(“A1”) is an Object Referring to Range “A1”. Cells (2, 1) or Range (“A2”) is an Object Referring to Range “A2”.

VBA – Properties, Methods and Events.

- Properties are the Physical Characteristics of objects – For example Worksheets. Count, Worksheets. Visible = False, Range (“A1:B15”).Rows. Count, Range (“A1:A50”).Font. Bold = True.
- Methods are the Actions that Can be Performed by Objects or on Objects For Example Worksheets.Save, Worksheets.Calculate, Range(“A1:A50”).ClearContents, ActiveCell.CopySpecial.
- Objects Can Respond to Events, Such as Mouse Click, Double Click on a Cell, Active a Worksheet, Open/Close/Save a Workbook, etc.

VBA – Macro

- VBA Macro starts with ‘sub’ keyword and ends with ‘End Sub’
- The format is as follows –

Sub Any_Name ()

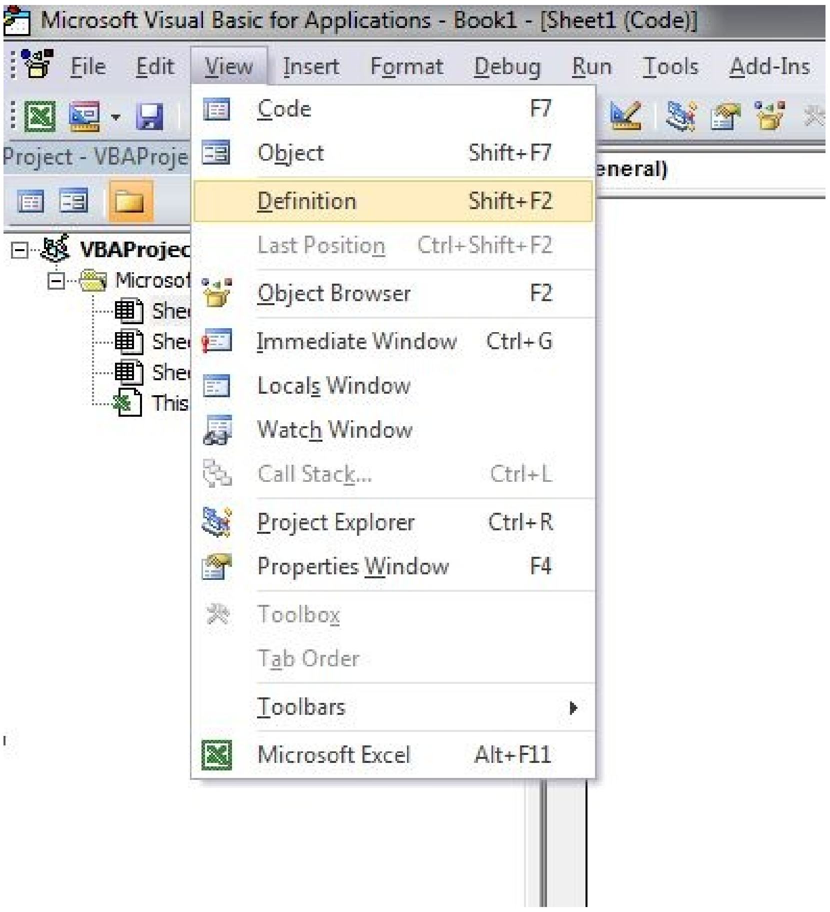
‘Your Code here in between

End Sub

- VBA macro can be assigned to any form control for example a button or it can be executed independently via some command option assigned to it.
- VBA macro can also be executed from the Macros window in the view tab of the menu bar.

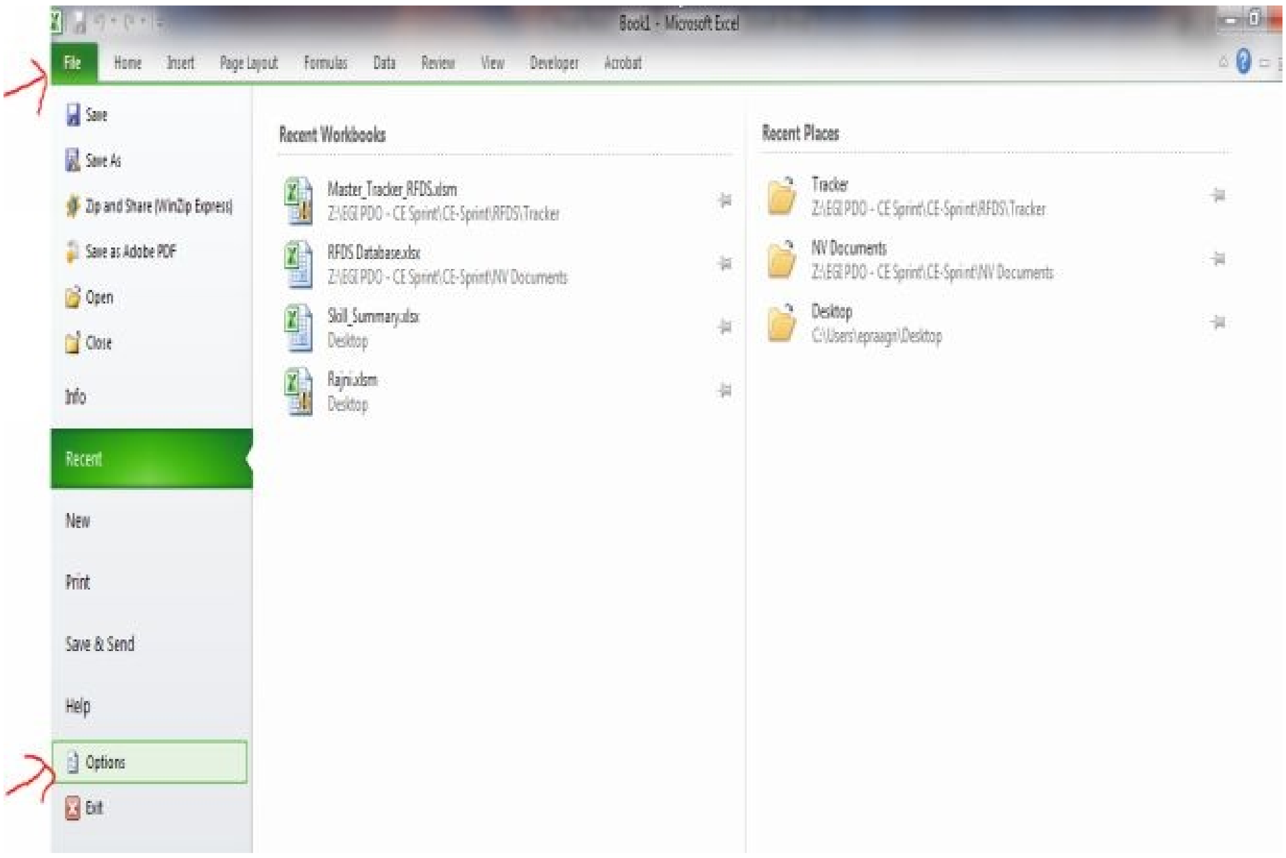
- Once VBA code is written in the excel file it should be saved with .xlsm extension.
- Some VBA Shortcuts are as follows –
    - ALT+F11- To view VBA Editor
    - ALT+F8- To display all macros
    - ALT+Q- To close VBA Editor and return to Excel
    - F5- To run a Macro
    - F2- Display Object Browser
    - F7- Display code editor
    - F1- Display help
    - Ctrl+G – Immediate Window
    - F4 – Properties window
    - Ctrl+R – Project Explorer.

See below for the shortcuts described above.

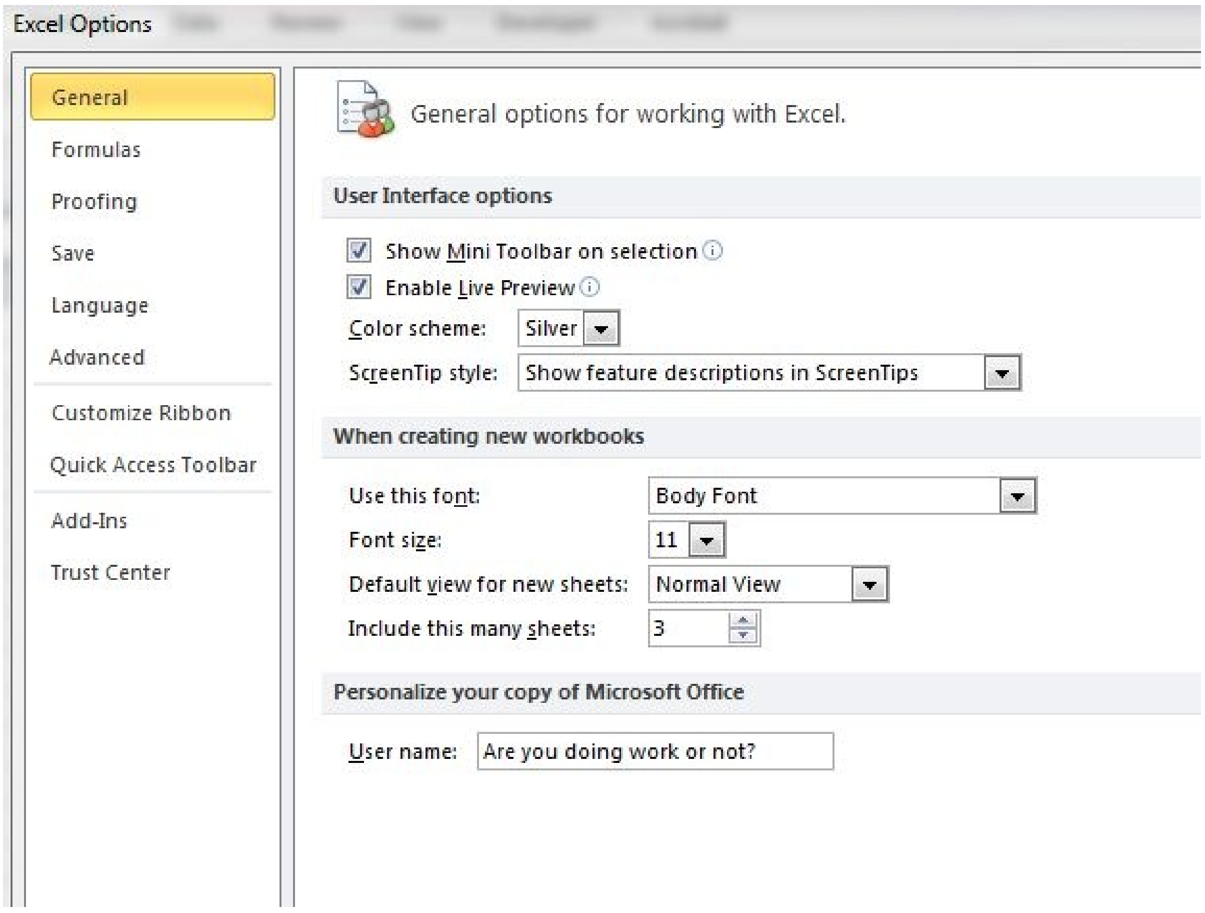


Get Started –

1.) Enable Developer Tab First – Go To File Tab->



2.) Go To Options-> A Window will appear like this ->



General options for working with Excel.

User Interface options

- [✓] Show Mini Toolbar on selection
- [✓] Enable Live Preview

Color scheme: Silver

ScreenTip style: Show feature descriptions in ScreenTips

When creating new workbooks

Use this font: Body Font

Font size: 11

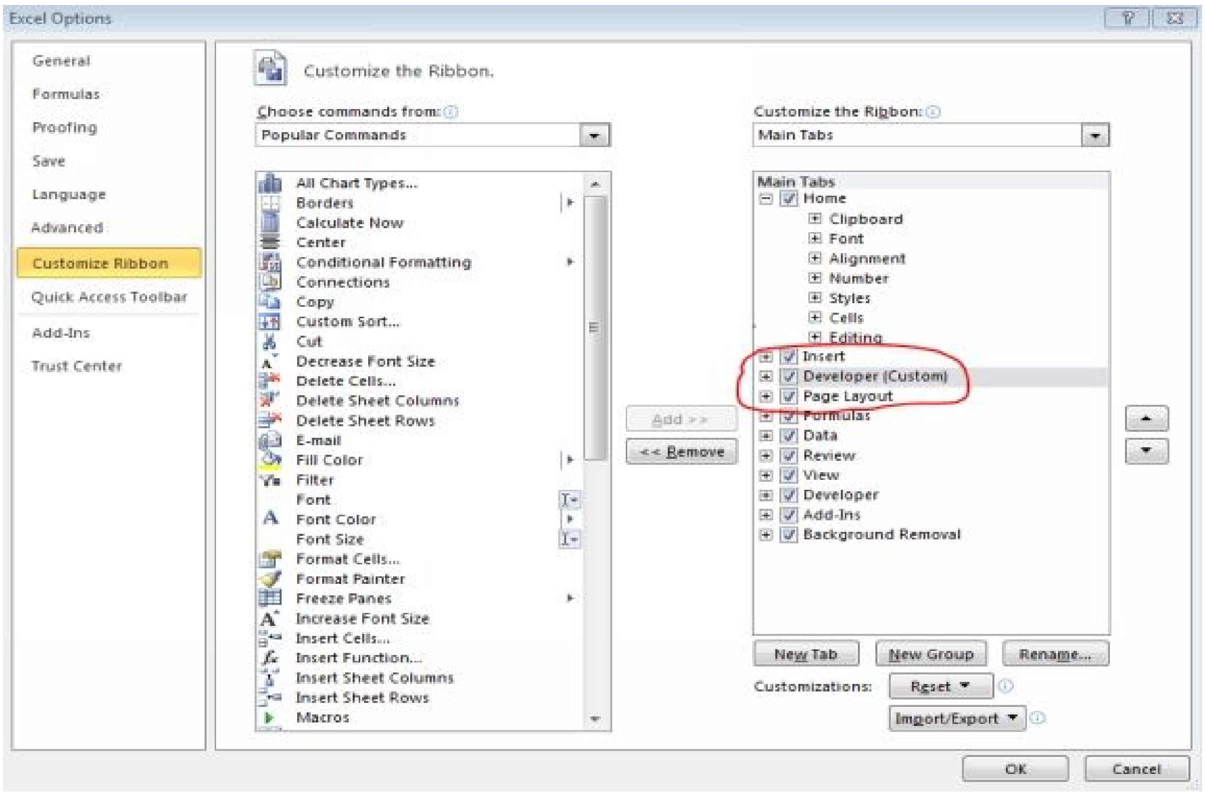
Default view for new sheets: Normal View

Include this many sheets: 3

Personalize your copy of Microsoft Office

User name: Are you doing work or not?

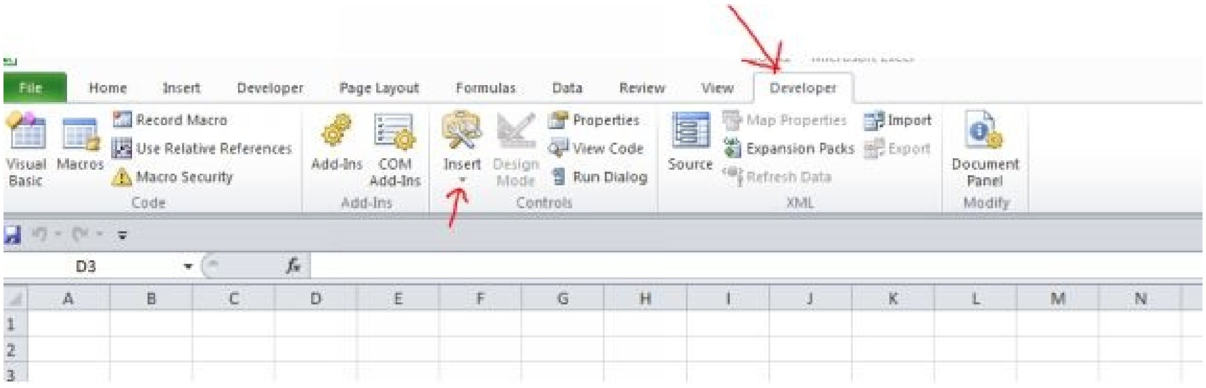
3.) Go To Customize Ribbon and **check** Developer Tab as shown below and then press OK->



Choose commands from: Popular Commands

Customize the Ribbon: Main Tabs

4.) Developer tab will be shown as below on your excel sheet. The developer tab appears on the menu bar.

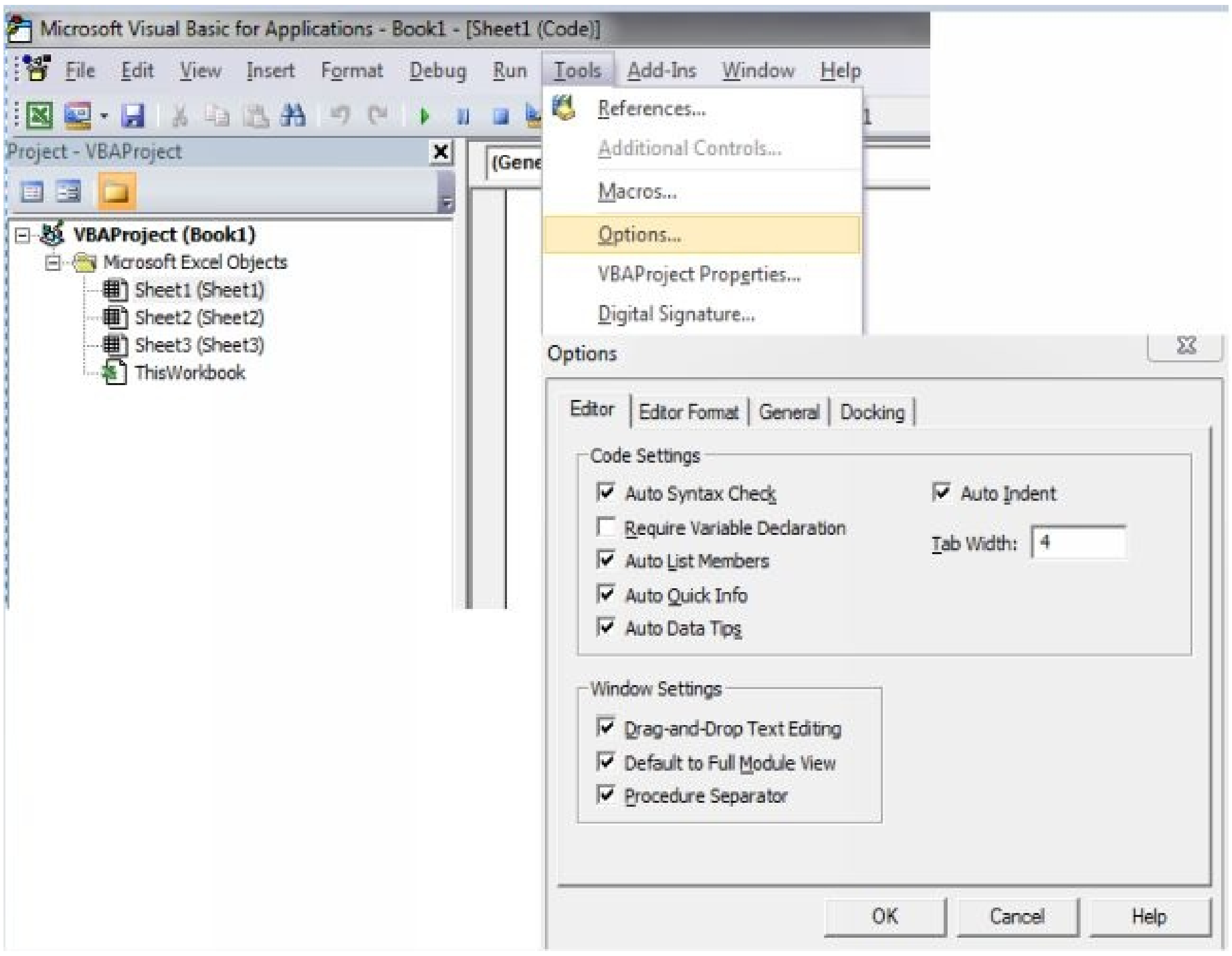


<u>Before Creating your first macro make sure to do following things –</u>

1.) Press ALT+F11- VBA IDE will get opened.

2.) Now click Tools tab in the menu bar on the top of the IDE. Select Options from the drop down as shown below in the snapshot.

3.) A window will appear. Just check whether the things shown in the snapshot below are checked or not.



Code Settings

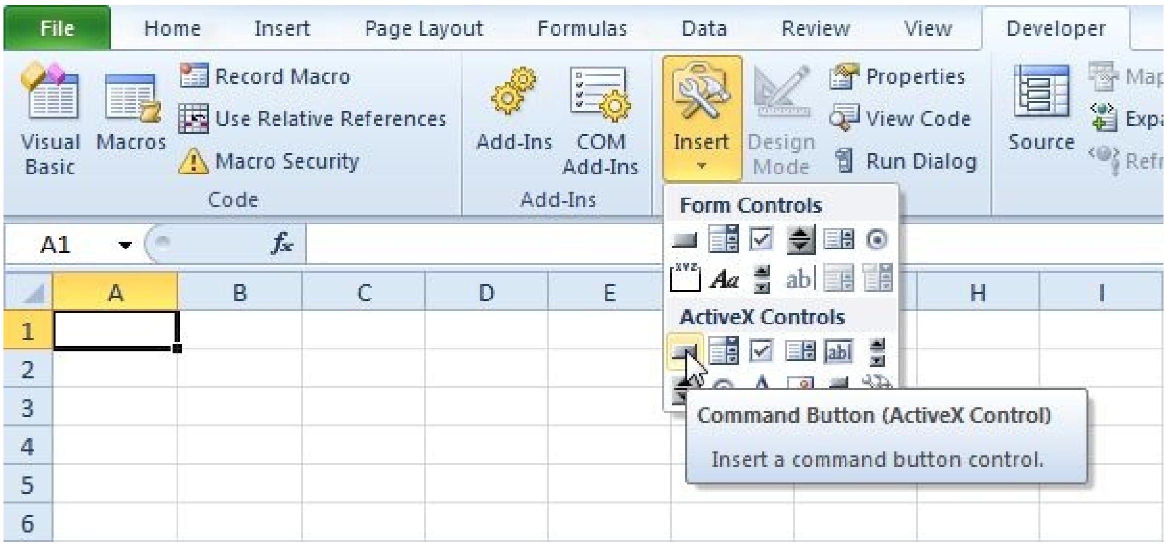
- [✓] Auto Syntax Check
- [ ] Require Variable Declaration
- [✓] Auto List Members
- [✓] Auto Quick Info
- [✓] Auto Data Tips
- [✓] Auto Indent
- Tab Width: 4

Window Settings

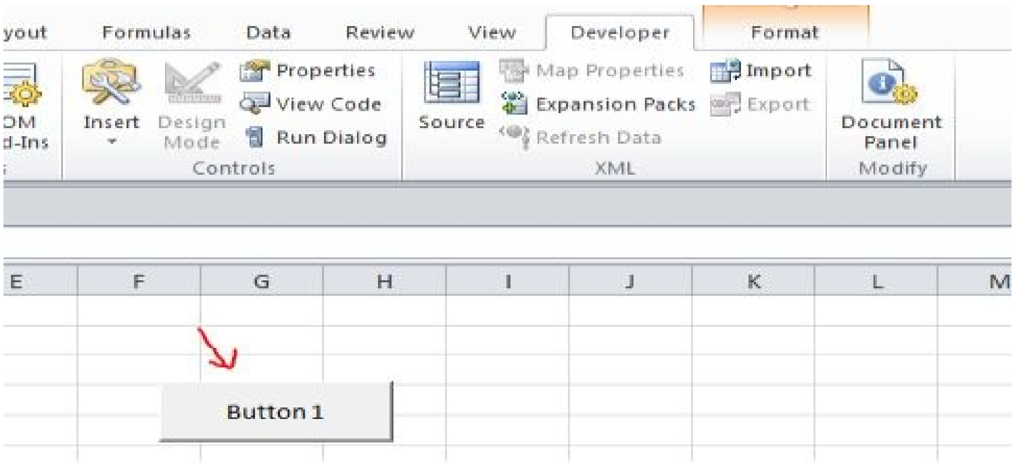
- [✓] Drag-and-Drop Text Editing
- [✓] Default to Full Module View
- [✓] Procedure Separator

Let Us Write Our First VBA Code by inserting Form control.

1.) Once you have developer tab on your menu bar click it and then go to Insert inside the developer tab and click it as shown below.



2.) Form controls window appears as shown above, Now click on the button and Insert as shown below.



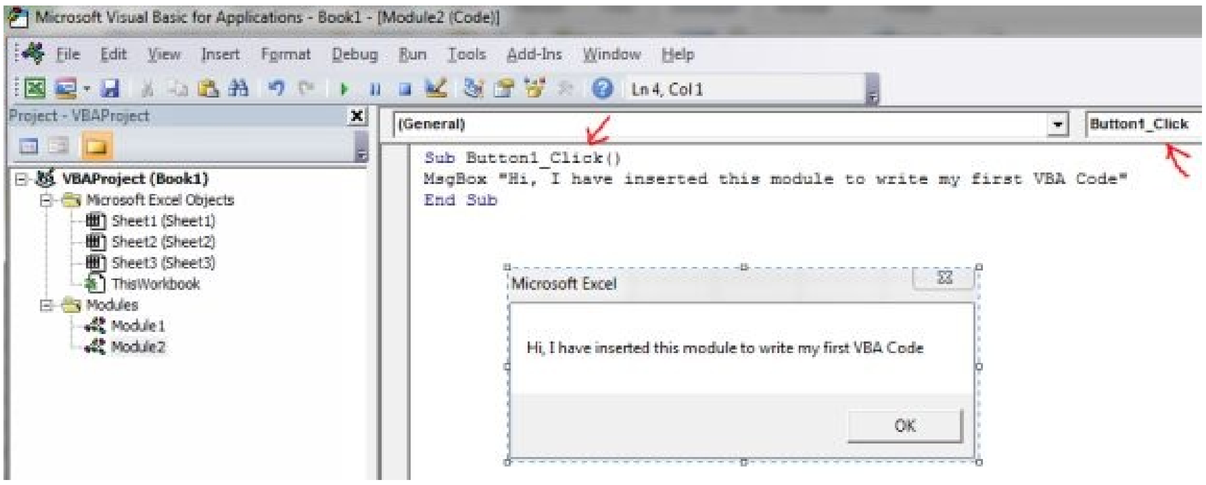
3.) Right Click the Button1 and assign a new macro. VBA IDE will get opened. Write a simple VBA Code is as follows –

Sub Button1_Click ()

MsgBox "Hi, I have inserted this module to write my first VBA Code"

End Sub

4.) See below for the code in code window and the output in the message box that appears.

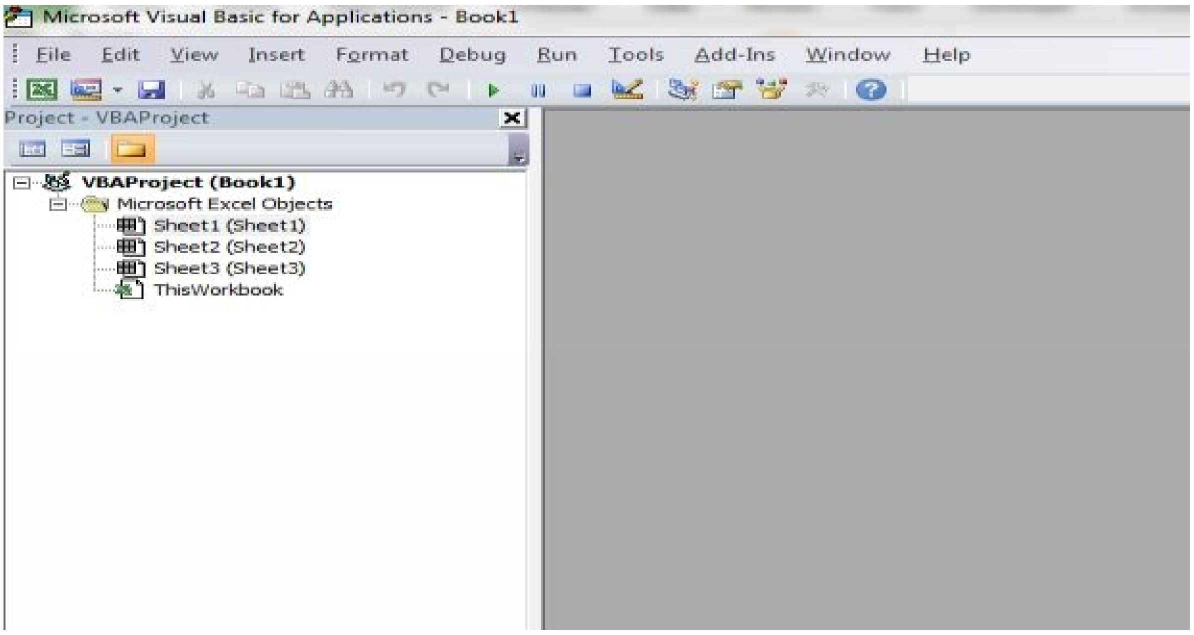


```
Sub Button1_Click()
MsgBox "Hi, I have inserted this module to write my first VBA Code"
End Sub
```

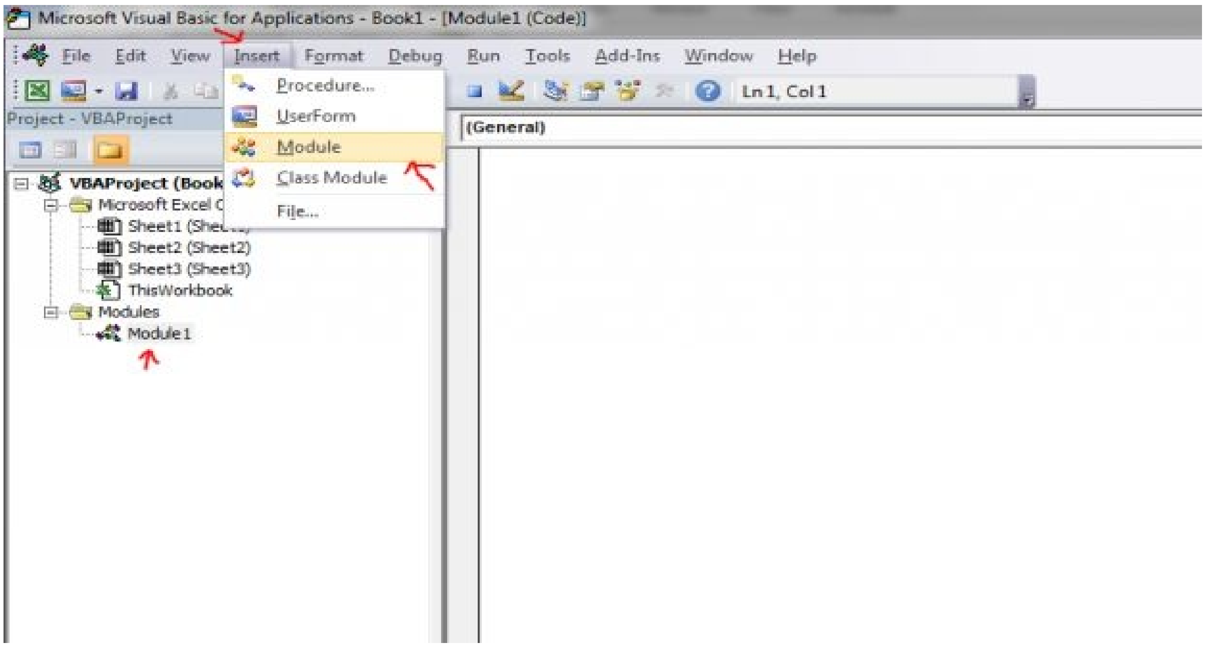
```
Hi, I have inserted this module to write my first VBA Code
```

How to access VBA Editor without inserting form control –

1.) Press ALT+F11 in your excel file. A window will open as shown below.



2.) Now Insert a module in it as shown below. Module 1 is the name of the first module inserted as shown below with red arrows.



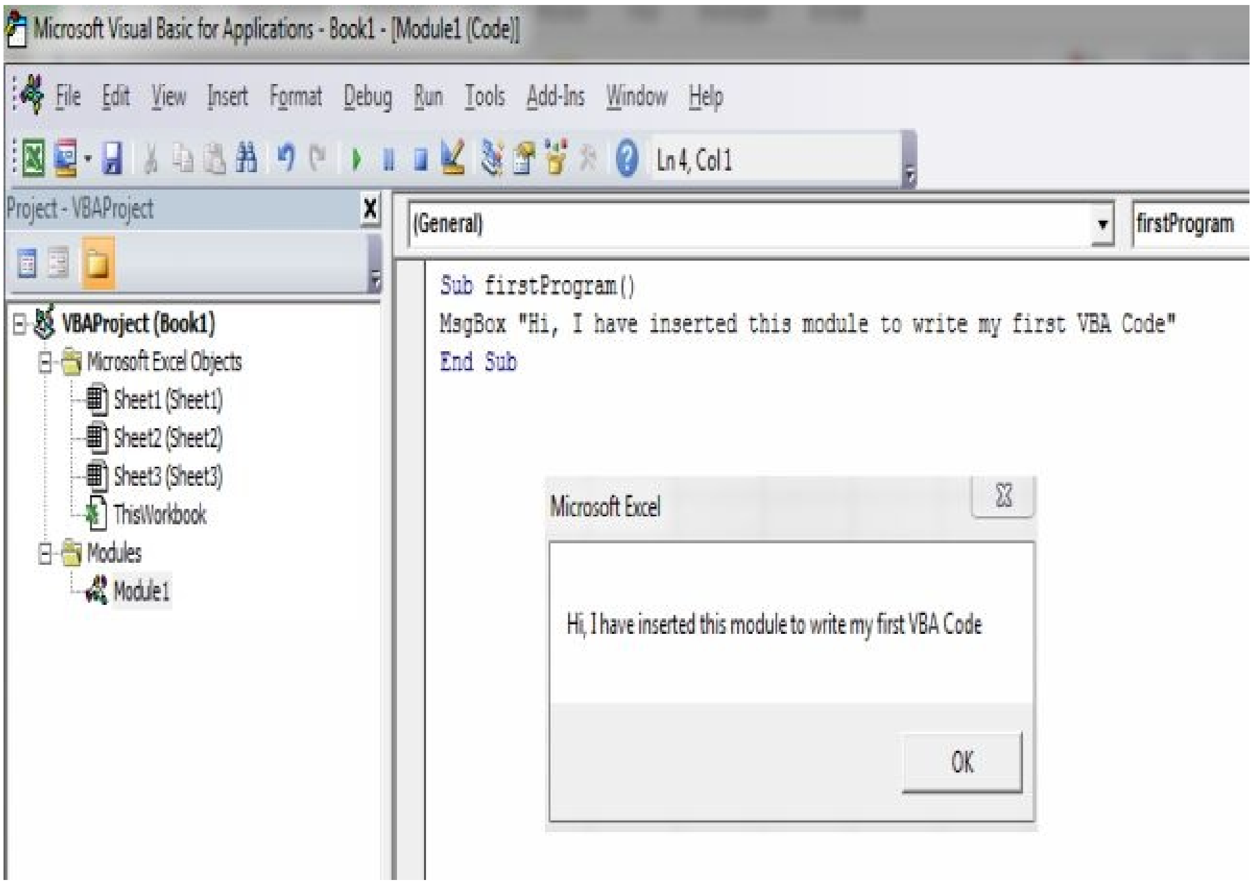
3.) Now in the above module window write a piece of VBA code as follows.

Sub first_Program ()

MsgBox "Hi, I have inserted this module to write my first VBA Code"

End Sub

See below for the code in code window and the output in the message box that appears.

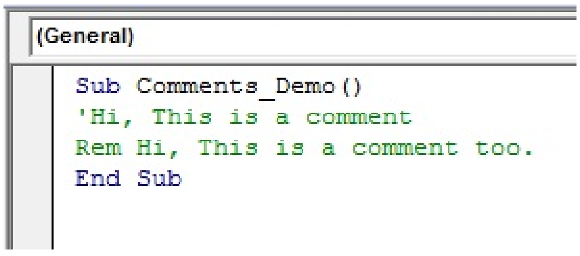


```
Sub firstProgram()
MsgBox "Hi, I have inserted this module to write my first VBA Code"
End Sub
```

```
Hi, I have inserted this module to write my first VBA Code
```

The Comments in VBA

- Statements or sentences in VBA code starting with single quote or REM keyword is a comment. See below in VBA IDE.



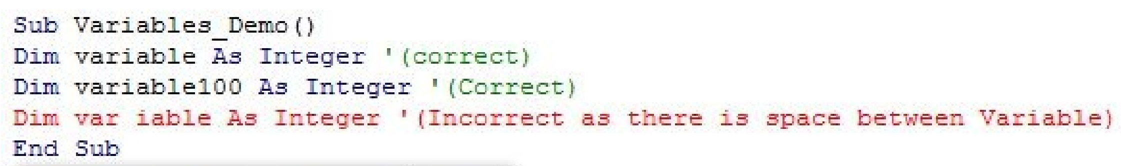
```
Sub Comments_Demo()
'Hi, This is a comment
Rem Hi, This is a comment too.
End Sub
```

The Concept of Variables

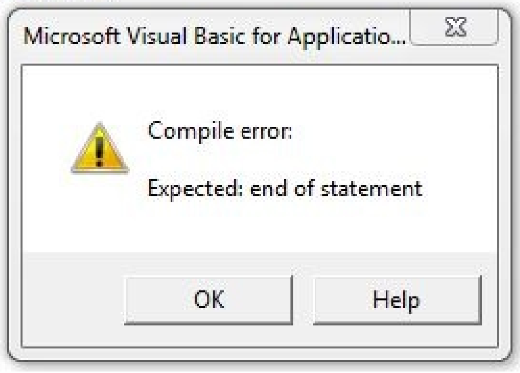
The following are the rules when naming the variables in Excel VBA-

- They must not exceed 40 characters
- They must contain only letters, numbers and underscore characters
- No spacing is allowed
- It must not begin with a number
- Examples – var1, my_var etc.

See below in VBA IDE



```
Sub Variables_Demo()
Dim variable As Integer '(correct)
Dim variable100 As Integer '(Correct)
Dim var iable As Integer '(Incorrect as there is space between Variable)
End Sub
```



Microsoft Visual Basic for Applicatio...

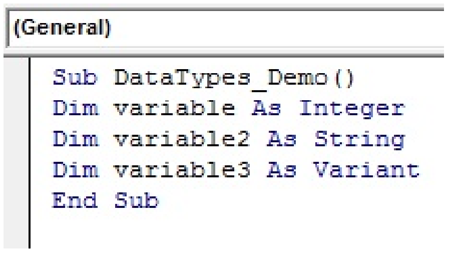
Compile error:

Expected: end of statement

Data Types

1.) Numeric

2.) Non Numeric (String, Date, Boolean, Object, variant)



(General)

```
Sub DataTypes_Demo()
Dim variable As Integer
Dim variable2 As String
Dim variable3 As Variant
End Sub
```

Declaration of variables

1.) Implicit (Variable is initialized and is variant by default)

2.) Explicit (Example – Dim var as Integer)

Conditional and Logical Operators

1.) Conditional – (=, <, >, >=, <=, <> etc.)

2.) Logical (AND, OR, XOR and NOT)

Control Structures

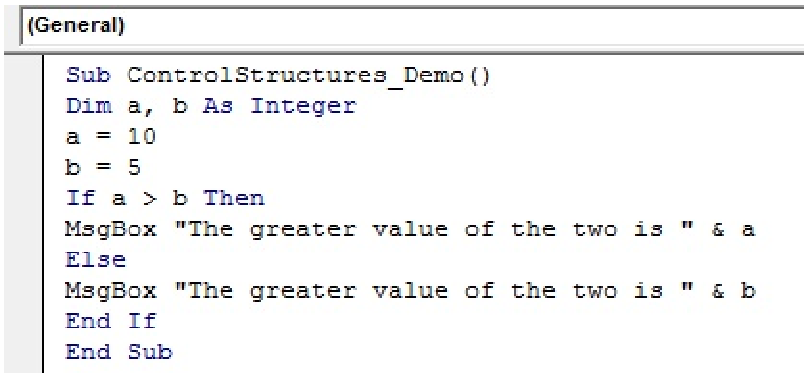
1.) If then Else

```
1. If Index = 0 Then
           MsgBox 'Hi Zero'
   Else If Index = 1 Then
           MsgBox 'Hi One'
   Else
           MsgBox 'Hi None'
   End If
```

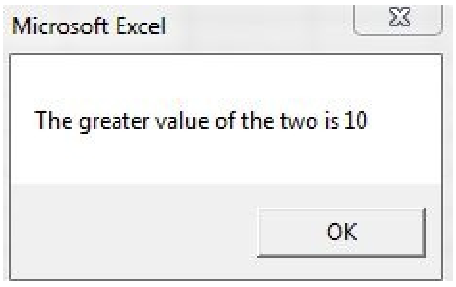
2.) Switch case

```
1. Select Case Index
           Case 0
                   MsgBox 'Hi Zero'
           Case 1
                   MsgBox 'Hi One'
           Case Else
                   MsgBox 'Hi None
   End Select
```

See below in VBA IDE



```
Sub ControlStructures_Demo()
Dim a, b As Integer
a = 10
b = 5
If a > b Then
MsgBox "The greater value of the two is " & a
Else
MsgBox "The greater value of the two is " & b
End If
End Sub
```



Microsoft Excel

The greater value of the two is 10

Loop Structures in VBA

1.) The loop structures are used when repetitive work is being done.

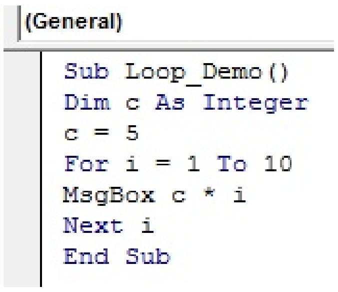
2.) Loops make the execution sequentially given the jump. For example if a process is required to be run 10 times then we use 1 to 10. If a process is required to be done only odd times then we fo 1 to 10 in steps of 2.

3.) There are various kinds of loop statements in VBA.

1. For Next Loop
2. While Wend Loop
3. Do while loop
4. Etc

4.) See below for code in IDE.

Displaying table of 5



```
Sub Loop_Demo()
Dim c As Integer
c = 5
For i = 1 To 10
MsgBox c * i
Next i
End Sub
```

Working with worksheets

1.) As already explained earlier everything in VBA is an object for example worksheet, Range, Cell etc.

2.) Worksheets (“Sheet1”).cells (1,2).value refers to the B1 in Sheet1 of excel workbook.

Error Handling – Following are the ways for it

1.) On Error GoTo LineNumber (Goes to the specified line number and Code resumes from)

2.) On Error GoTo 0 (Disables enabled error handler in the current procedure and resets it to Nothing.)

3.) On Error GoTo -1 (Disables enabled exception in the current procedure and resets it to Nothing.)

4.) On Error resume Next (Control goes to the statement immediately after the errored statement)

<u>Arrays Manipulation in VBA</u>

1.) Arrays are a significant part of any programming language which acts as front end processing database.

2.) Unlike variables Arrays are used to store multiple values of same data type. Arrays store homogeneous data.

3.) Array stores the values on the basis of key value pair i.e. array indexes the values stores in it. The Lower Bound and Upper bound can be set manually for an array.

4.) If you try to access any index of the already built array that does not exist an error named “Subscript out of Range” occurs.

5.) Arrays can be of two types which are Static and Dynamic. The former is used when the size of an array remains the same throughout in the procedure while the latter permits the user to regulate the size of an array at run time.

6.) Mostly in VBA single dimensional array is used although VBA provides the functionality of multi-dimensional arrays.

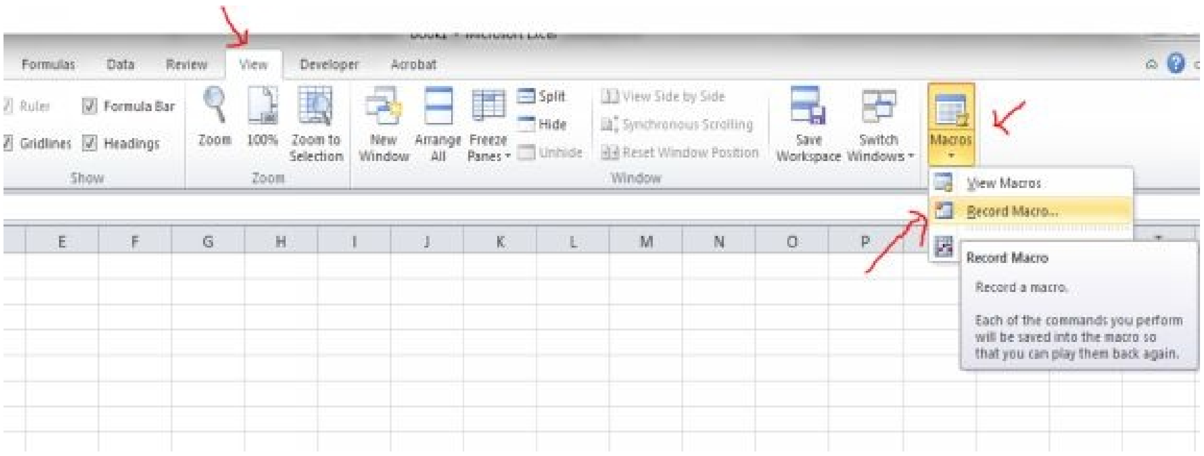
7.) A user can only pass an array to a procedure using By Reference and a user can return an array from a function but the array, it is assigned to, must not be currently allocated.

8.) Any worksheet in a workbook has data in the form of Rows and columns. So basically the data stored is two dimensional. Any data from excel sheet can be directly transferred to a two dimensional array and vice versa for manipulation purposes.

<u>Recording a VBA Macro</u>

1.) Let’s record a macro.

2.) Go to excel while you are working on. Click View in menu bar and then click Macros on the extreme right. A drop down will appear. Click on the Record Macro just below View Macros. See below.



3.) Now once you have clicked the record macro, whatever activity you will do in that excel file will be recorded.

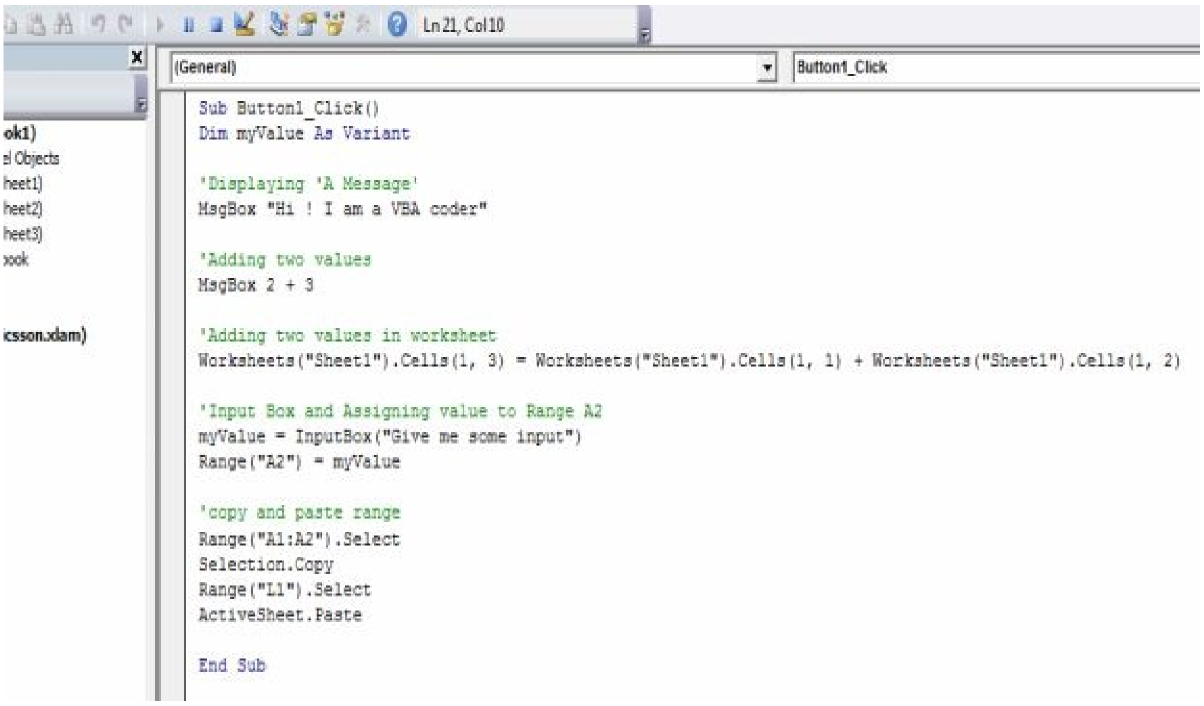
4.) The Macro recording is useful when you are doing some repetitive task on the daily basis.

5.) The Macro recorded can be tweaked further as per the user’s need.

An Example VBA

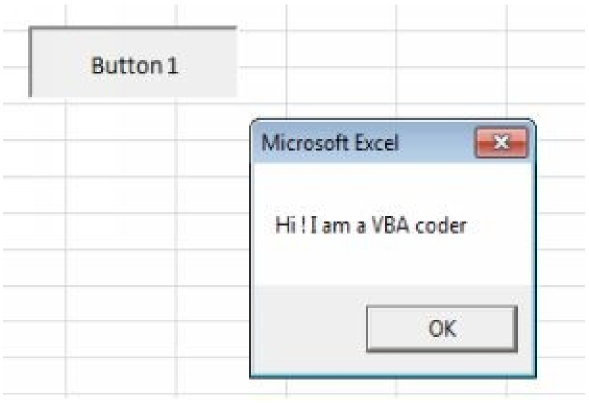
```
Sub Button1_Click ()
Dim myValue As Variant
‘Displaying ‘A Message’
MsgBox “Hi! I am a VBA coder”
‘Adding two values
MsgBox 2 + 3
‘Adding two values in worksheet
Worksheets (“Sheet1”).Cells (1, 3) =Worksheets (“Sheet1”).Cells (1, 1)+Worksheets
(“Sheet1”).Cells (1, 2)
‘Input Box and Assigning value to Range A2
MyValue = InputBox(“Give me some input”)
Range (“A2”) = MyValue
‘Copy and paste range
Range (“A1:A2”).Select
Selection.Copy
Range (“L1”).Select
ActiveSheet.Paste
End Sub
```

Code in IDE – VBA Window.

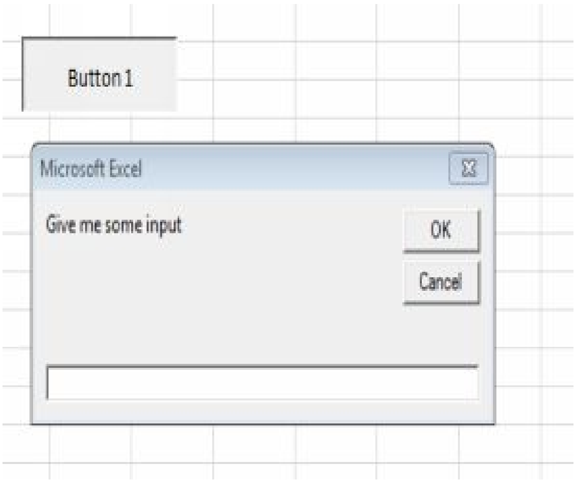


```
Sub Button1_Click()
Dim myValue As Variant

'Displaying 'A Message'
MsgBox "Hi ! I am a VBA coder"

'Adding two values
MsgBox 2 + 3

'Adding two values in worksheet
Worksheets("Sheet1").Cells(1, 3) = Worksheets("Sheet1").Cells(1, 1) + Worksheets("Sheet1").Cells(1, 2)

'Input Box and Assigning value to Range A2
myValue = InputBox("Give me some input")
Range("A2") = myValue

'copy and paste range
Range("A1:A2").Select
Selection.Copy
Range("L1").Select
ActiveSheet.Paste

End Sub
```

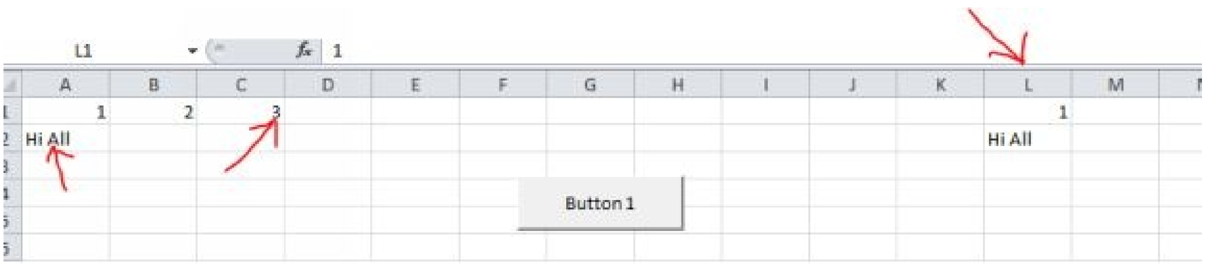
Output is as follows -



```
Hi ! I am a VBA coder
```



```
Give me some input
```

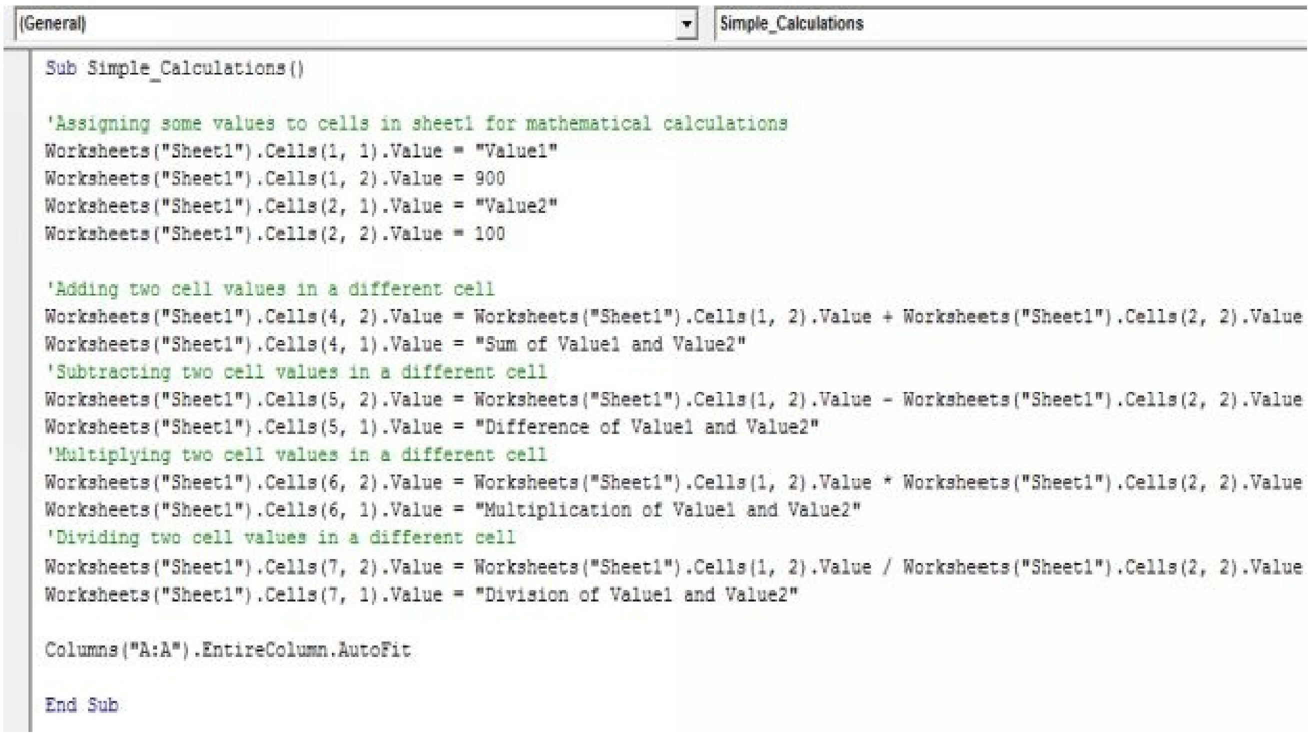


	A	B	C	D	E	F	G	H	I	J	K	L	M
1	1	2	3									1	
2	Hi All											Hi All	

SAMPLE VBA CODES FOR YOUR USAGE –

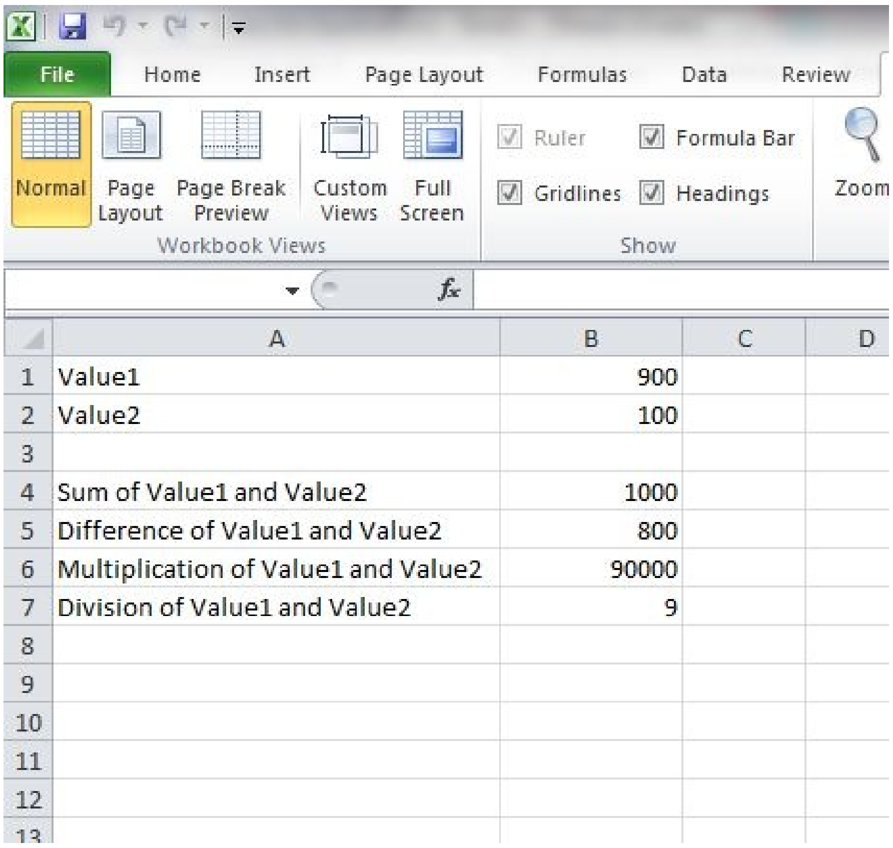
1.) VBA code for simple mathematical calculations using values in worksheet's cells. Some code lines are as follows –

1. Worksheets(“Sheet1”).cells(1,1).value = 900
2. Worksheets(“Sheet1”).cells(1,1).value =100
3. Worksheets(“Sheet1”).Cells(4, 2).Value = Worksheets(“Sheet1”).Cells(1, 2).Value + Worksheets(“Sheet1”).Cells(2, 2).Value
4. Complete code is shown below in the snapshot.



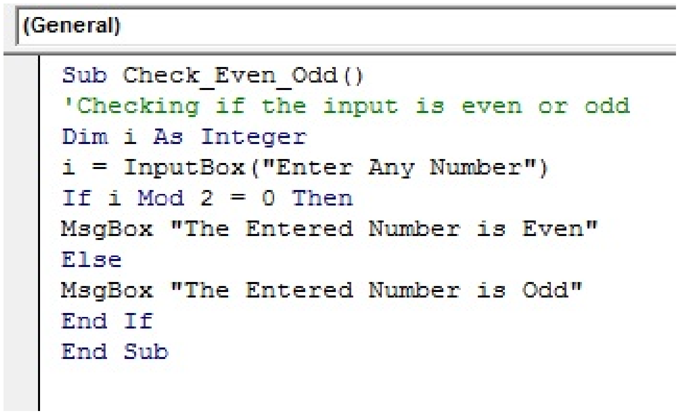
```
Sub Simple_Calculations()

'Assigning some values to cells in sheet1 for mathematical calculations
Worksheets("Sheet1").Cells(1, 1).Value = "Value1"
Worksheets("Sheet1").Cells(1, 2).Value = 900
Worksheets("Sheet1").Cells(2, 1).Value = "Value2"
Worksheets("Sheet1").Cells(2, 2).Value = 100

'Adding two cell values in a different cell
Worksheets("Sheet1").Cells(4, 2).Value = Worksheets("Sheet1").Cells(1, 2).Value + Worksheets("Sheet1").Cells(2, 2).Value
Worksheets("Sheet1").Cells(4, 1).Value = "Sum of Value1 and Value2"
'Subtracting two cell values in a different cell
Worksheets("Sheet1").Cells(5, 2).Value = Worksheets("Sheet1").Cells(1, 2).Value - Worksheets("Sheet1").Cells(2, 2).Value
Worksheets("Sheet1").Cells(5, 1).Value = "Difference of Value1 and Value2"
'Multiplying two cell values in a different cell
Worksheets("Sheet1").Cells(6, 2).Value = Worksheets("Sheet1").Cells(1, 2).Value * Worksheets("Sheet1").Cells(2, 2).Value
Worksheets("Sheet1").Cells(6, 1).Value = "Multiplication of Value1 and Value2"
'Dividing two cell values in a different cell
Worksheets("Sheet1").Cells(7, 2).Value = Worksheets("Sheet1").Cells(1, 2).Value / Worksheets("Sheet1").Cells(2, 2).Value
Worksheets("Sheet1").Cells(7, 1).Value = "Division of Value1 and Value2"

Columns("A:A").EntireColumn.AutoFit

End Sub
```

Output is as follows –



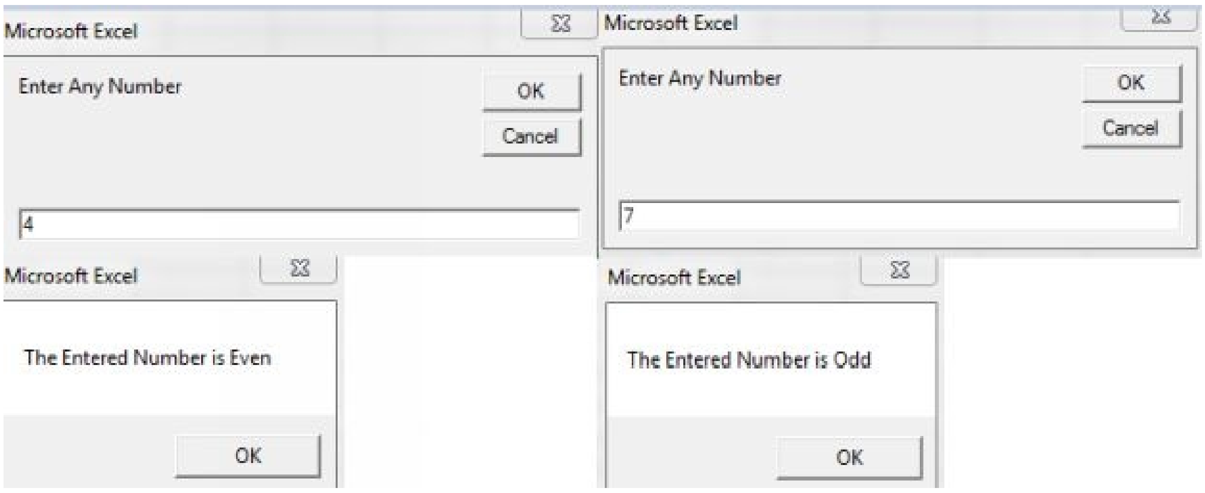
	A	B	C	D
1	Value1	900		
2	Value2	100		
3				
4	Sum of Value1 and Value2	1000		
5	Difference of Value1 and Value2	800		
6	Multiplication of Value1 and Value2	90000		
7	Division of Value1 and Value2	9		
8				
9				
10				
11				
12				

2.) Checking if a Number entered by the user is Even or Odd. See Code –



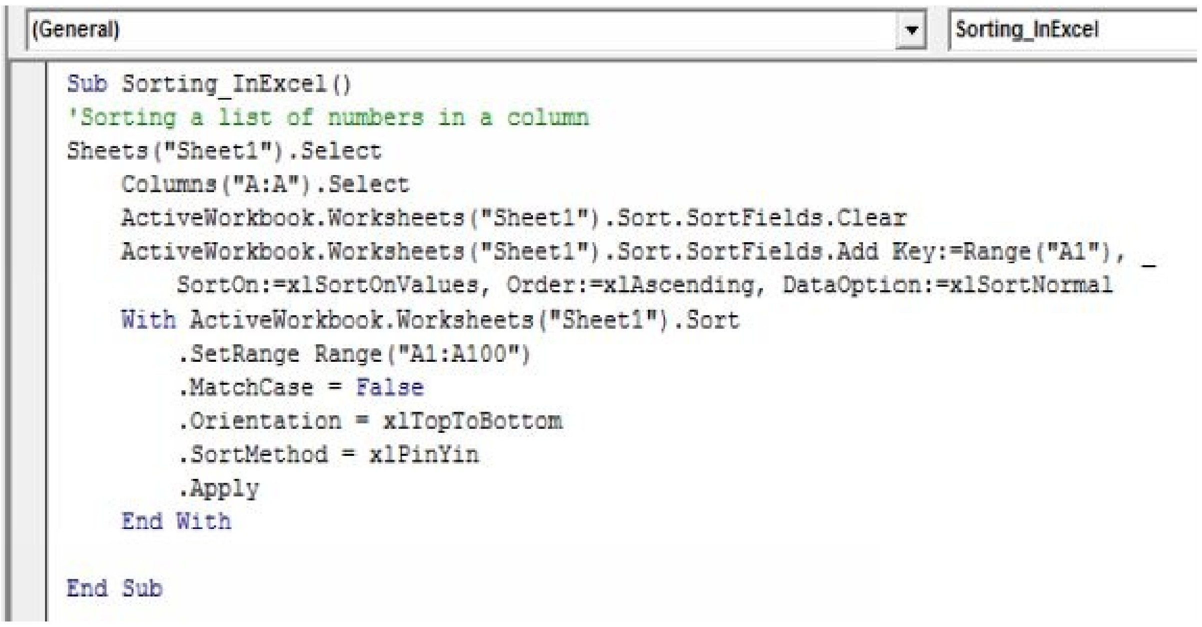
```
(General)
Sub Check_Even_Odd()
'Checking if the input is even or odd
Dim i As Integer
i = InputBox("Enter Any Number")
If i Mod 2 = 0 Then
MsgBox "The Entered Number is Even"
Else
MsgBox "The Entered Number is Odd"
End If
End Sub
```

Output –



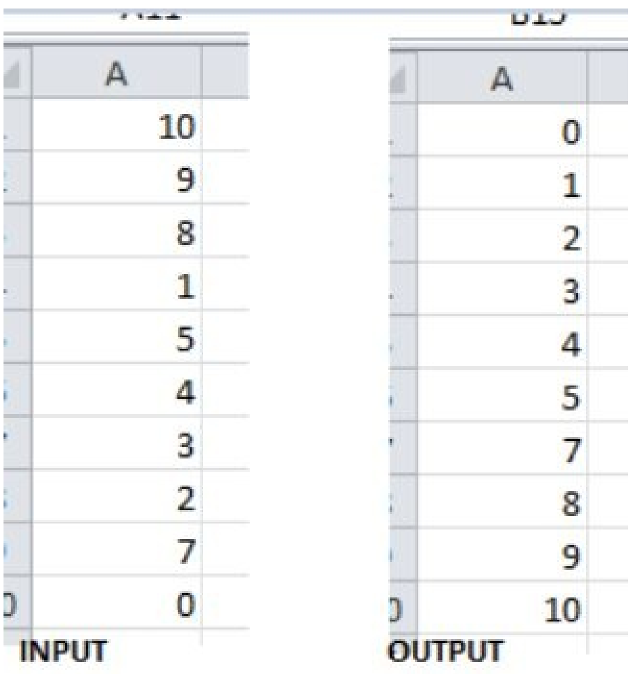
```
Microsoft Excel
Enter Any Number
4

Microsoft Excel
Enter Any Number
7

Microsoft Excel
The Entered Number is Even

Microsoft Excel
The Entered Number is Odd
```

3.) Code for Sorting a column –



```
(General)                                    Sorting_InExcel
Sub Sorting_InExcel()
'Sorting a list of numbers in a column
Sheets("Sheet1").Select
    Columns("A:A").Select
    ActiveWorkbook.Worksheets("Sheet1").Sort.SortFields.Clear
    ActiveWorkbook.Worksheets("Sheet1").Sort.SortFields.Add Key:=Range("A1"), _
        SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
    With ActiveWorkbook.Worksheets("Sheet1").Sort
        .SetRange Range("A1:A100")
        .MatchCase = False
        .Orientation = xlTopToBottom
        .SortMethod = xlPinYin
        .Apply
    End With

End Sub
```

Output –



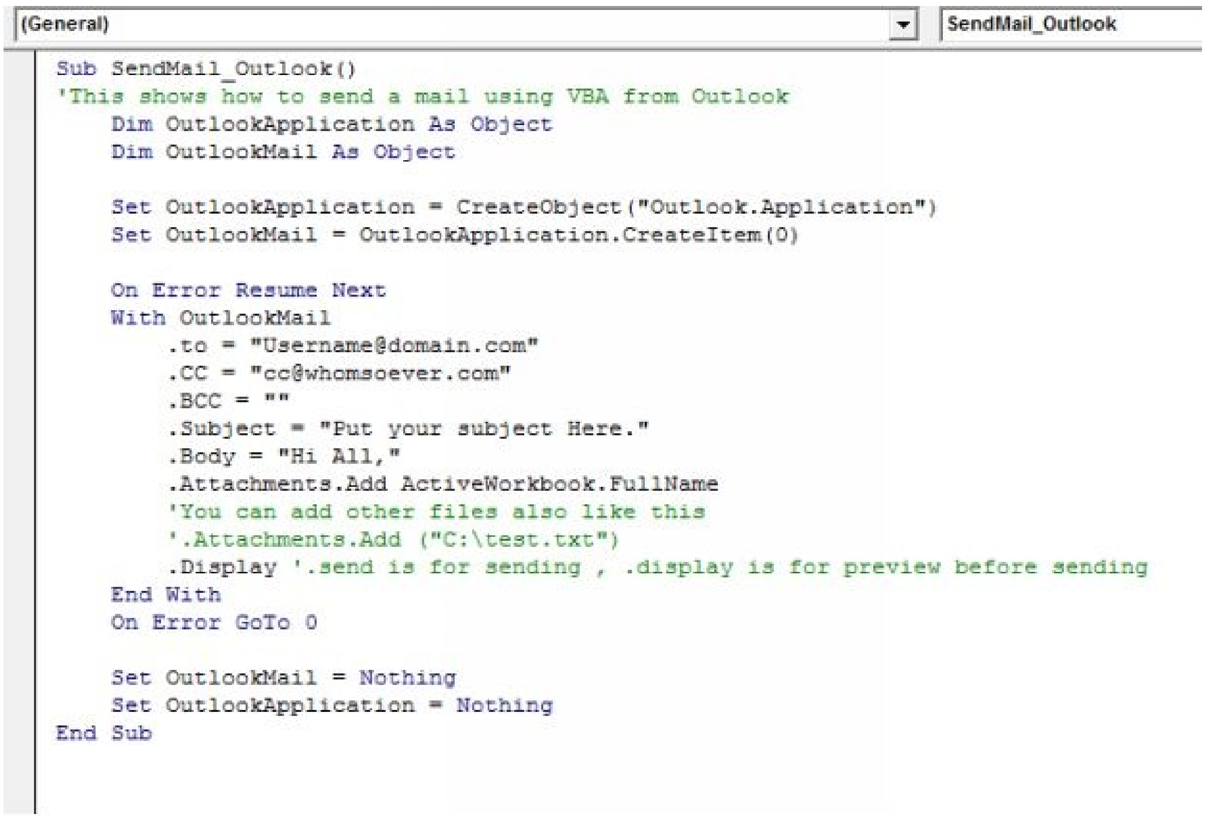
A
10
9
8
1
5
4
3
2
7
0

INPUT

A
0
1
2
3
4
5
7
8
9
10

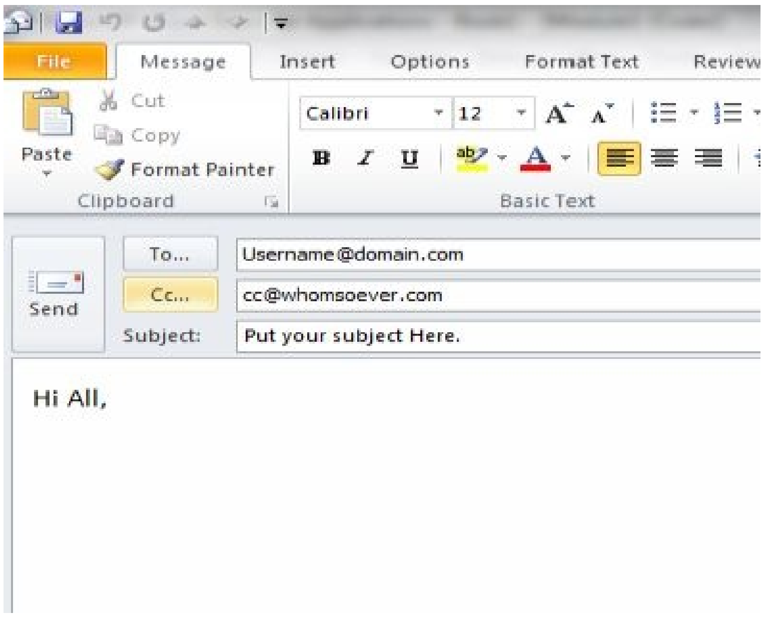
OUTPUT

4.) Code to send a mail from Excel via Outlook. Just paste the code as shown in the snapshot below and change your fields accordingly.



```
Sub SendMail_Outlook()
'This shows how to send a mail using VBA from Outlook
    Dim OutlookApplication As Object
    Dim OutlookMail As Object

    Set OutlookApplication = CreateObject("Outlook.Application")
    Set OutlookMail = OutlookApplication.CreateItem(0)

    On Error Resume Next
    With OutlookMail
        .to = "Username@domain.com"
        .CC = "cc@whomsoever.com"
        .BCC = ""
        .Subject = "Put your subject Here."
        .Body = "Hi All,"
        .Attachments.Add ActiveWorkbook.FullName
        'You can add other files also like this
        '.Attachments.Add ("C:\test.txt")
        .Display '.send is for sending , .display is for preview before sending
    End With
    On Error GoTo 0

    Set OutlookMail = Nothing
    Set OutlookApplication = Nothing
End Sub
```

Output –



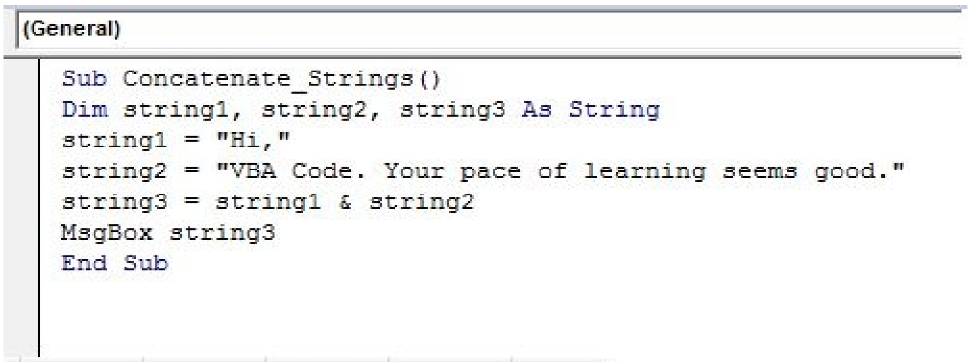
To... Username@domain.com

Cc... cc@whomsoever.com

Subject: Put your subject Here.

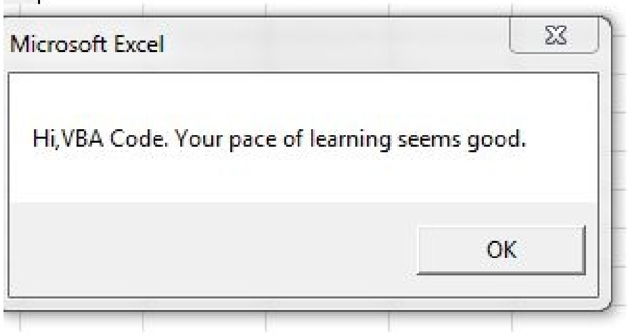
Hi All,

5.) Concatenating Two Strings in VBA. See code below-



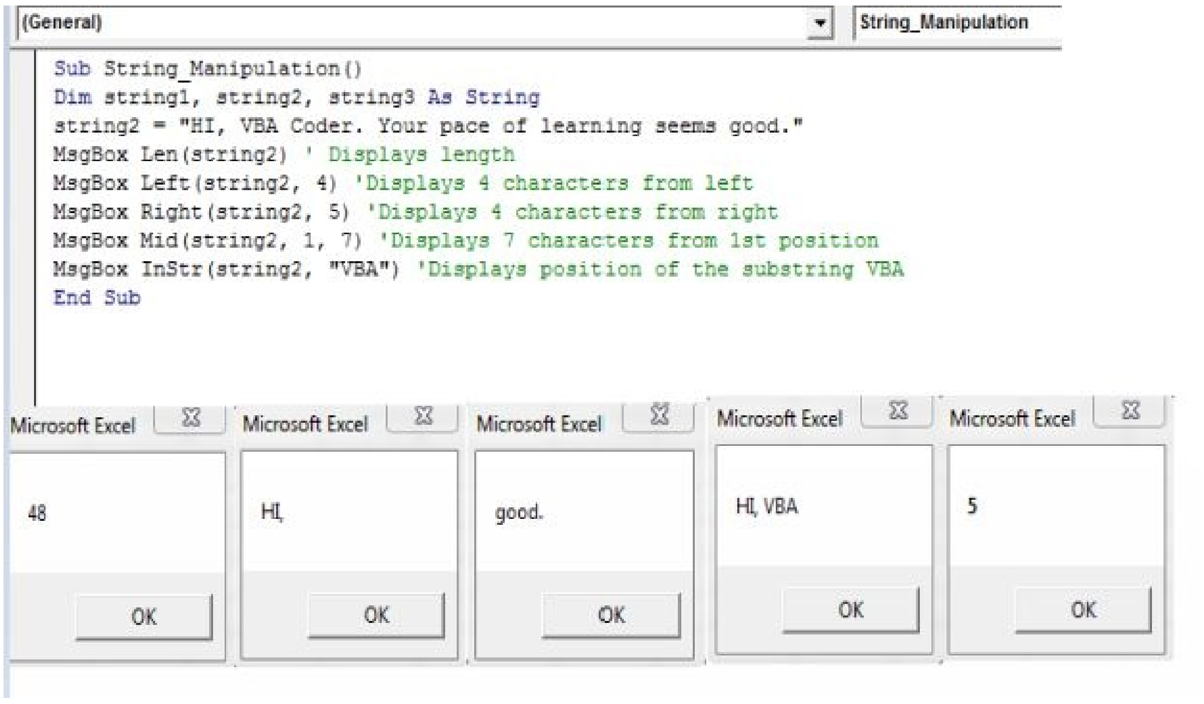
(General)

```
Sub Concatenate_Strings()
Dim string1, string2, string3 As String
string1 = "Hi,"
string2 = "VBA Code. Your pace of learning seems good."
string3 = string1 & string2
MsgBox string3
End Sub
```



```
Hi,VBA Code. Your pace of learning seems good.
```

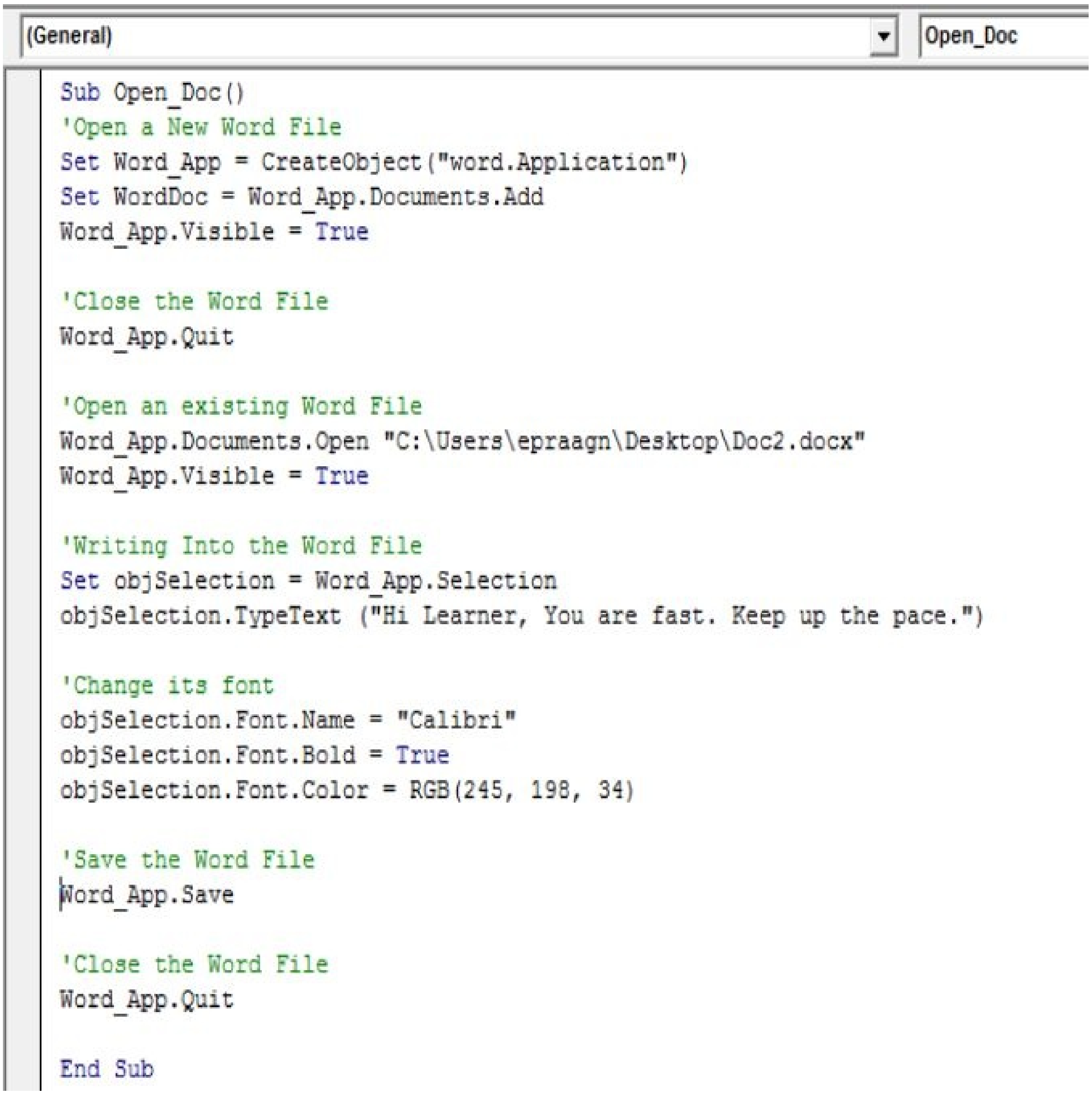
6.) Some String manipulation Functions in VBA. See code and output as shown below.


(General) | String_Manipulation

```
Sub String_Manipulation()
Dim string1, string2, string3 As String
string2 = "HI, VBA Coder. Your pace of learning seems good."
MsgBox Len(string2) ' Displays length
MsgBox Left(string2, 4) 'Displays 4 characters from left
MsgBox Right(string2, 5) 'Displays 4 characters from right
MsgBox Mid(string2, 1, 7) 'Displays 7 characters from 1st position
MsgBox InStr(string2, "VBA") 'Displays position of the substring VBA
End Sub
```

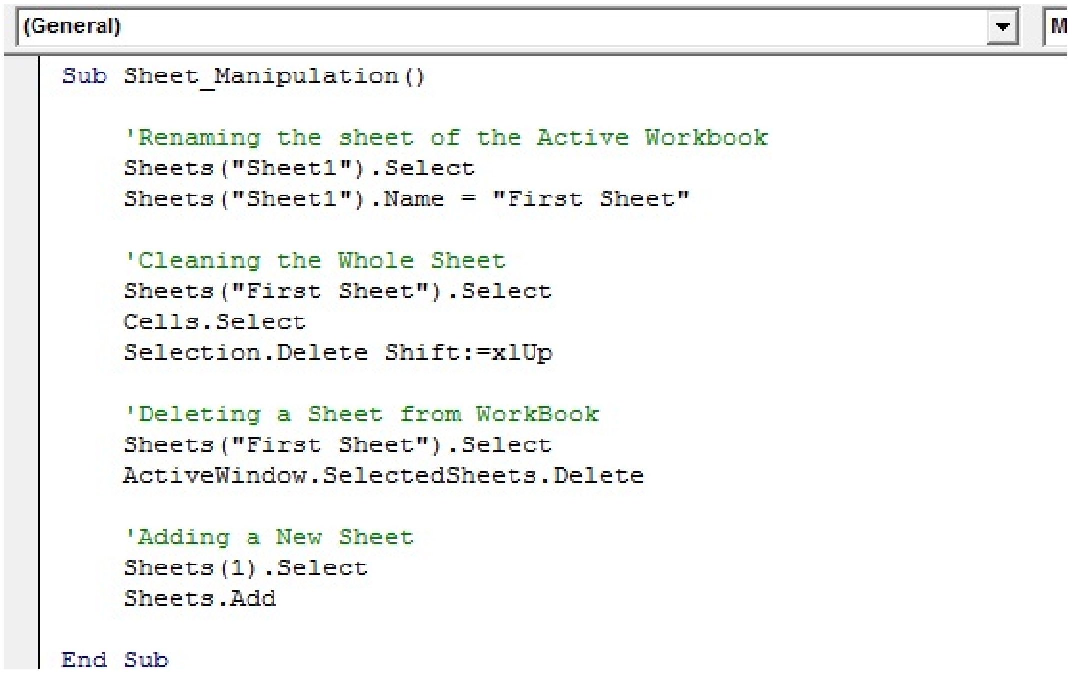
```
Microsoft Excel: 48
Microsoft Excel: HI,
Microsoft Excel: good.
Microsoft Excel: HI, VBA
Microsoft Excel: 5
```

7.) Working with Doc file –



```
Sub Open_Doc()
'Open a New Word File
Set Word_App = CreateObject("word.Application")
Set WordDoc = Word_App.Documents.Add
Word_App.Visible = True

'Close the Word File
Word_App.Quit

'Open an existing Word File
Word_App.Documents.Open "C:\Users\epraagn\Desktop\Doc2.docx"
Word_App.Visible = True

'Writing Into the Word File
Set objSelection = Word_App.Selection
objSelection.TypeText ("Hi Learner, You are fast. Keep up the pace.")

'Change its font
objSelection.Font.Name = "Calibri"
objSelection.Font.Bold = True
objSelection.Font.Color = RGB(245, 198, 34)

'Save the Word File
Word_App.Save

'Close the Word File
Word_App.Quit

End Sub
```

8.) Sheet Manipulation. Adding new Sheet, Deleting an existing sheet, renaming a sheet and cleaning the whole sheet etc.

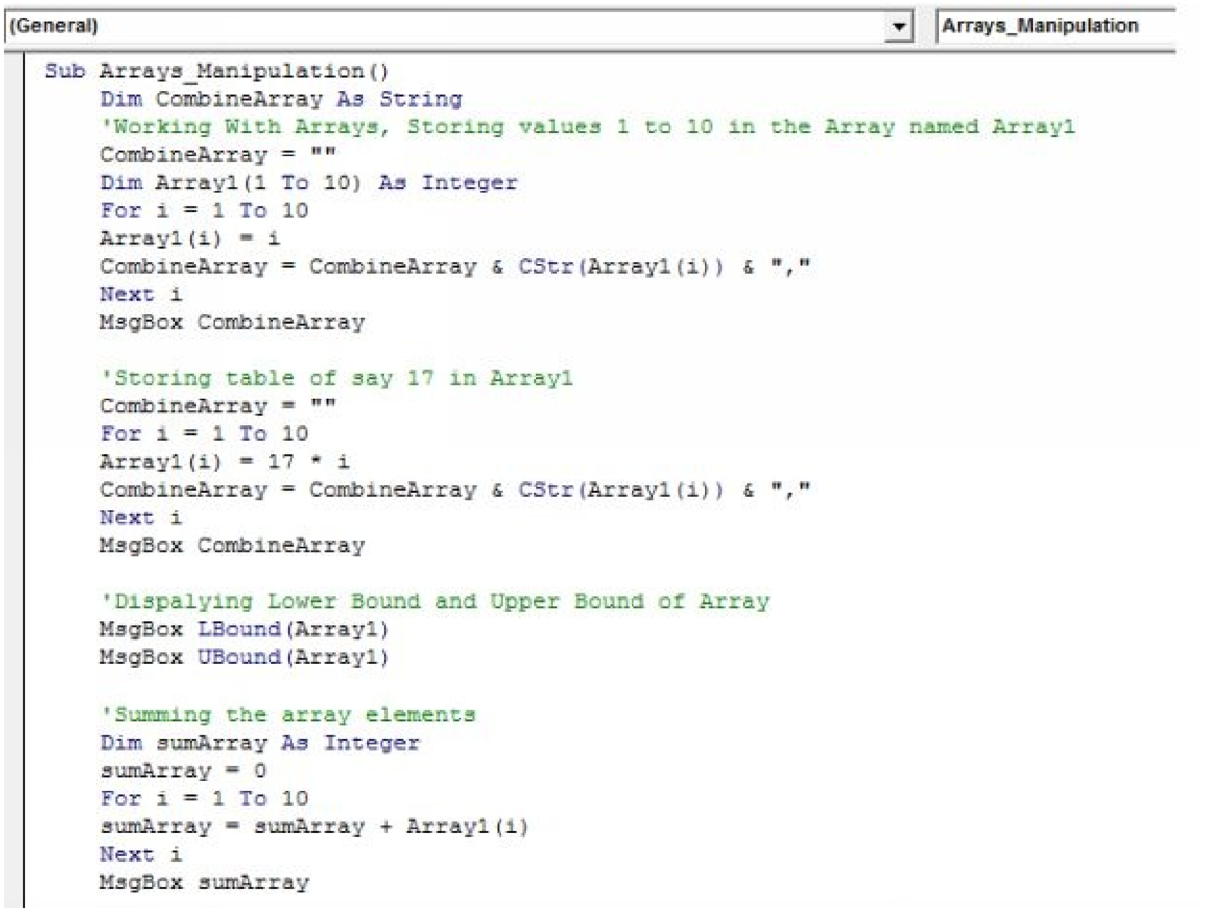


```
Sub Sheet_Manipulation()

    'Renaming the sheet of the Active Workbook
    Sheets("Sheet1").Select
    Sheets("Sheet1").Name = "First Sheet"

    'Cleaning the Whole Sheet
    Sheets("First Sheet").Select
    Cells.Select
    Selection.Delete Shift:=xlUp

    'Deleting a Sheet from WorkBook
    Sheets("First Sheet").Select
    ActiveWindow.SelectedSheets.Delete

    'Adding a New Sheet
    Sheets(1).Select
    Sheets.Add

End Sub
```

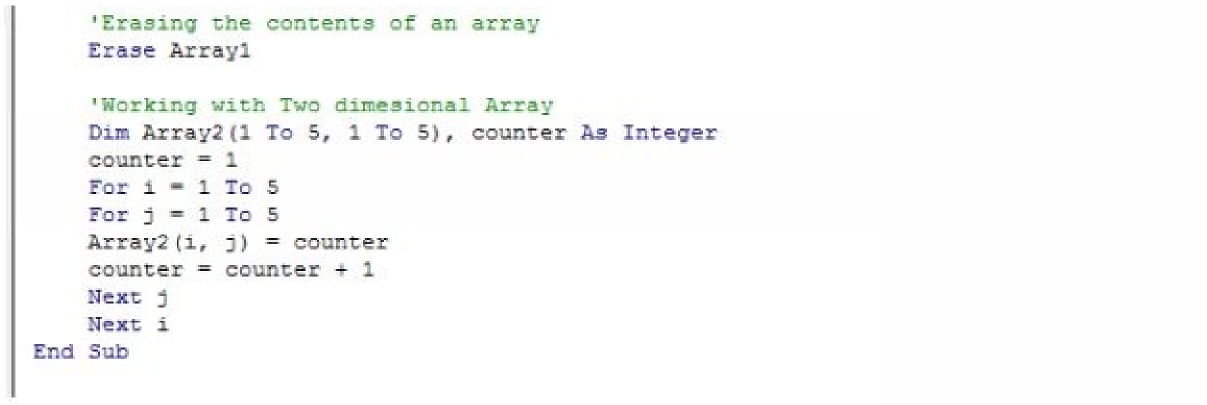
9.) Working with Arrays

```
Sub Arrays_Manipulation()
    Dim CombineArray As String
    ‘Working With Arrays, Storing values 1 to 10 in the Array named Array1
    CombineArray = ””
    Dim Array1(1 To 10) As Integer
    For i = 1 To 10
    Array1(i) = i
    CombineArray = CombineArray & CStr(Array1(i)) & “,”
    Next i
    MsgBox CombineArray
    ‘Storing table of say 17 in Array1
    CombineArray = ””
    For i = 1 To 10
    Array1(i) = 17 * i
    CombineArray = CombineArray & CStr(Array1(i)) & “,”
    Next i
    MsgBox CombineArray
    ‘Dispalying Lower Bound and Upper Bound of Array
    MsgBox LBound(Array1)
    MsgBox UBound(Array1)
    ‘Summing the array elements
    Dim sumArray As Integer
    sumArray = 0
    For i = 1 To 10
    sumArray = sumArray + Array1(i)
    Next i
    MsgBox sumArray
    ‘Erasing the contents of an array
    Erase Array1
    ‘Working with Two dimesional Array
    Dim Array2(1 To 5, 1 To 5), counter As Integer
    counter = 1
    For i = 1 To 5
    For j = 1 To 5
    Array2(i, j) = counter
    counter = counter + 1
    Next j
    Next i
End Sub
```

<u>**Code in VBA IDE**</u>



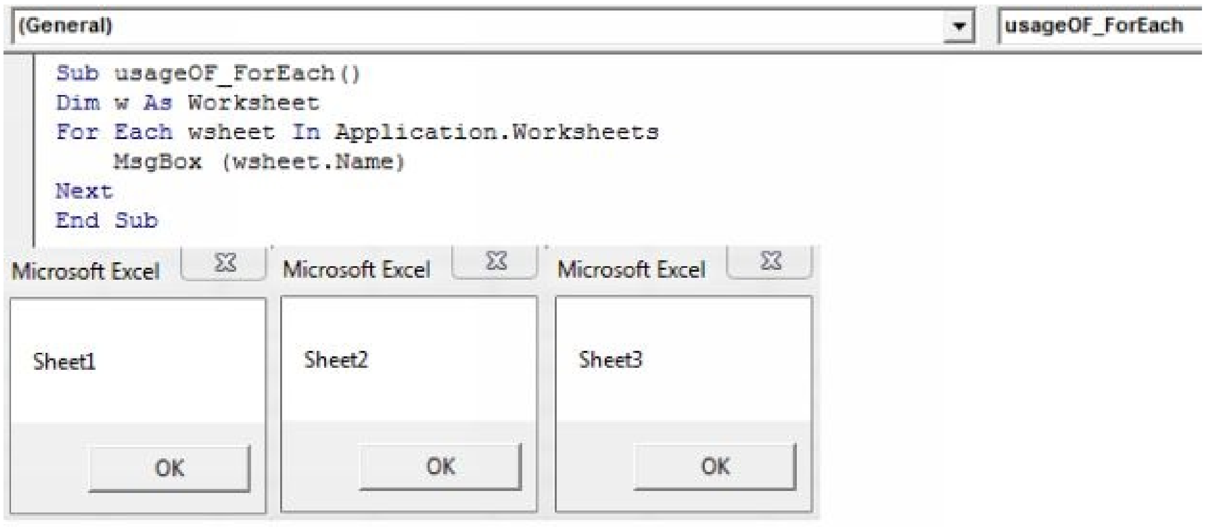
```
Sub Arrays_Manipulation()
    Dim CombineArray As String
    'Working With Arrays, Storing values 1 to 10 in the Array named Array1
    CombineArray = ""
    Dim Array1(1 To 10) As Integer
    For i = 1 To 10
    Array1(i) = i
    CombineArray = CombineArray & CStr(Array1(i)) & ","
    Next i
    MsgBox CombineArray

    'Storing table of say 17 in Array1
    CombineArray = ""
    For i = 1 To 10
    Array1(i) = 17 * i
    CombineArray = CombineArray & CStr(Array1(i)) & ","
    Next i
    MsgBox CombineArray

    'Dispalying Lower Bound and Upper Bound of Array
    MsgBox LBound(Array1)
    MsgBox UBound(Array1)

    'Summing the array elements
    Dim sumArray As Integer
    sumArray = 0
    For i = 1 To 10
    sumArray = sumArray + Array1(i)
    Next i
    MsgBox sumArray
```



```
    'Erasing the contents of an array
    Erase Array1

    'Working with Two dimesional Array
    Dim Array2(1 To 5, 1 To 5), counter As Integer
    counter = 1
    For i = 1 To 5
    For j = 1 To 5
    Array2(i, j) = counter
    counter = counter + 1
    Next j
    Next i
End Sub
```

The above code describes two kinds of arrays. Single dimensional and Double dimensional. It shows how to enter values in these kinds of arrays, how to access the elements of multi-dimensional array, how to perform operations like sum, sort, erase etc.

10.) Displaying Name of each work sheet and usage of For Each.


(General) usageOF_ForEach

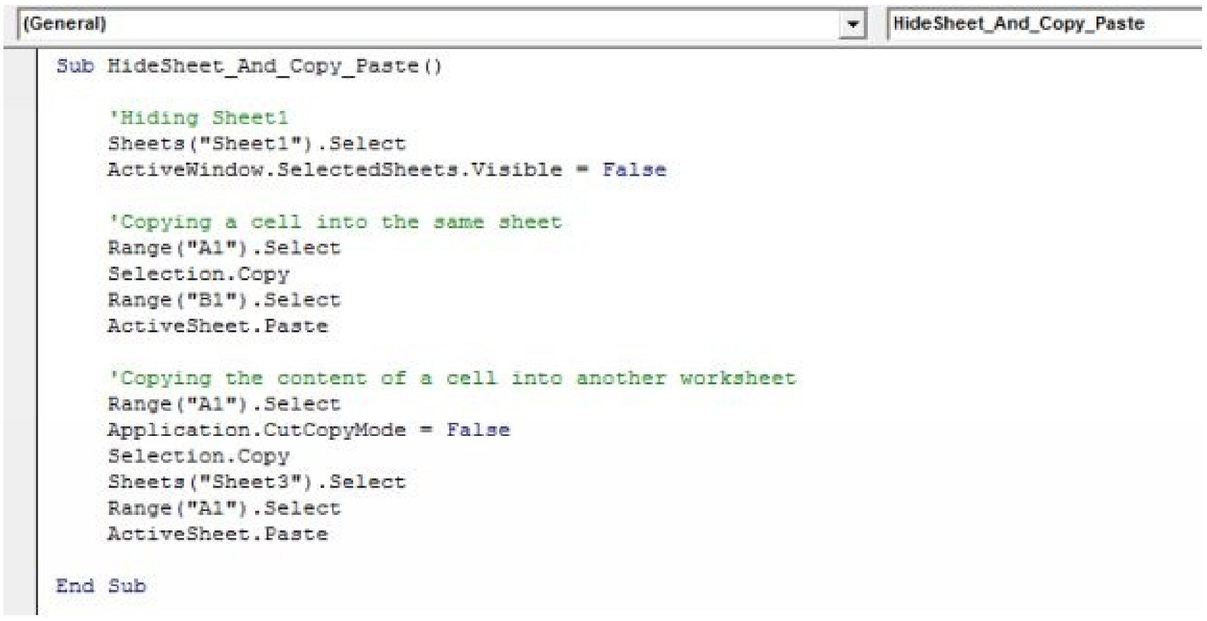
```
Sub usageOF_ForEach()
Dim w As Worksheet
For Each wsheet In Application.Worksheets
    MsgBox (wsheet.Name)
Next
End Sub
```

Microsoft Excel
Sheet1
OK

Microsoft Excel
Sheet2
OK

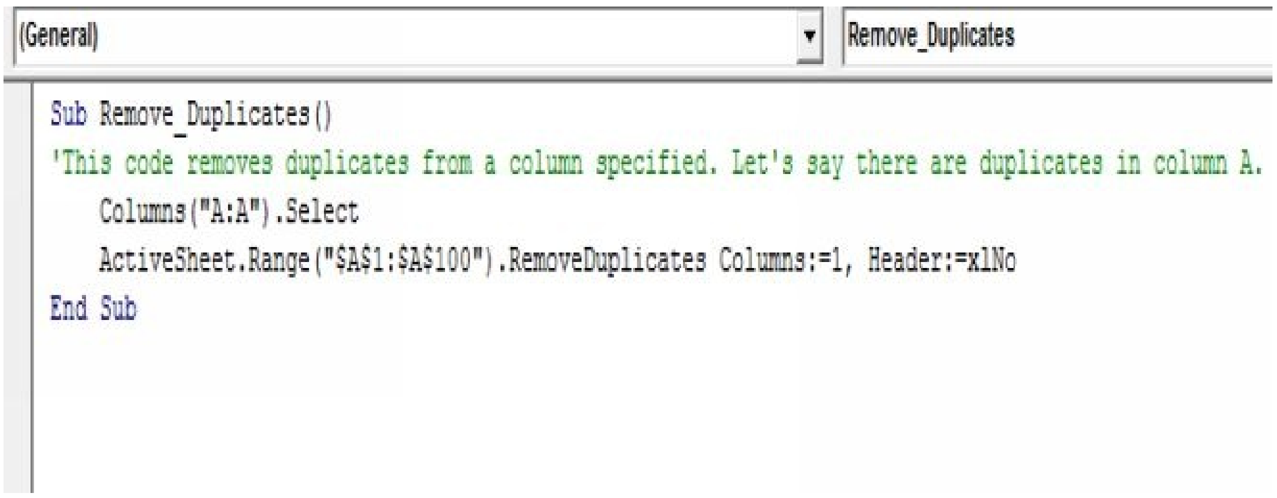
Microsoft Excel
Sheet3
OK

11.) Hiding a sheet, copy and paste into same sheet and another sheet.


(General) HideSheet_And_Copy_Paste

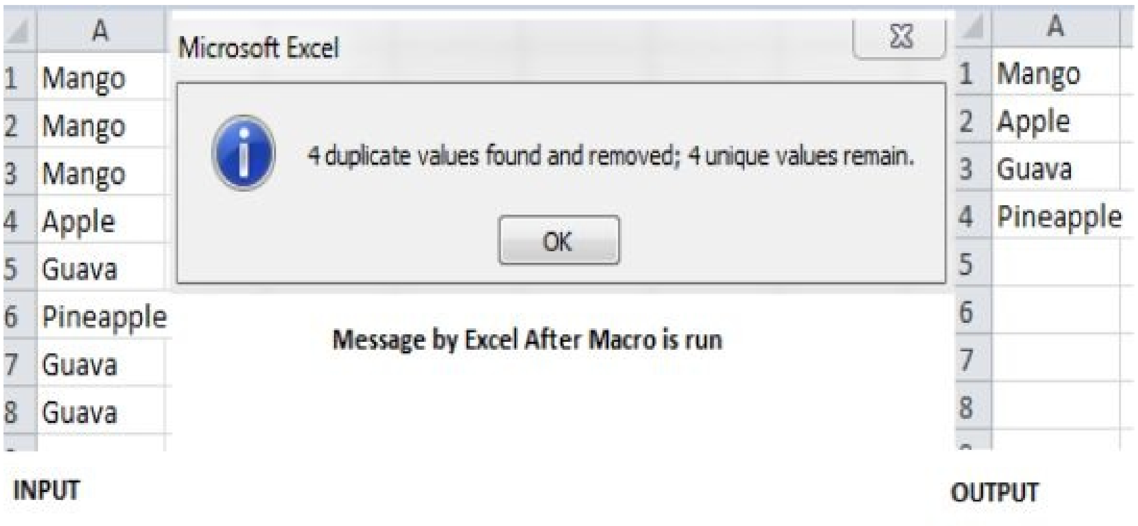
```
Sub HideSheet_And_Copy_Paste()

    'Hiding Sheet1
    Sheets("Sheet1").Select
    ActiveWindow.SelectedSheets.Visible = False

    'Copying a cell into the same sheet
    Range("A1").Select
    Selection.Copy
    Range("B1").Select
    ActiveSheet.Paste

    'Copying the content of a cell into another worksheet
    Range("A1").Select
    Application.CutCopyMode = False
    Selection.Copy
    Sheets("Sheet3").Select
    Range("A1").Select
    ActiveSheet.Paste

End Sub
```

12.) Removing duplicates from a column in excel worksheet.


(General) Remove_Duplicates

```
Sub Remove_Duplicates()
'This code removes duplicates from a column specified. Let's say there are duplicates in column A.
    Columns("A:A").Select
    ActiveSheet.Range("$A$1:$A$100").RemoveDuplicates Columns:=1, Header:=xlNo
End Sub
```

See the Input and Output as shown below –



	A
1	Mango
2	Mango
3	Mango
4	Apple
5	Guava
6	Pineapple
7	Guava
8	Guava

INPUT

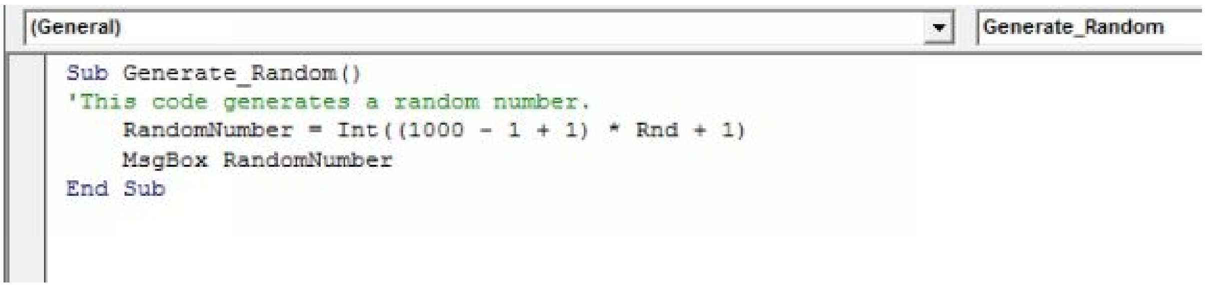
```
Microsoft Excel
4 duplicate values found and removed; 4 unique values remain.
OK
```

Message by Excel After Macro is run

	A
1	Mango
2	Apple
3	Guava
4	Pineapple
5	
6	
7	
8	

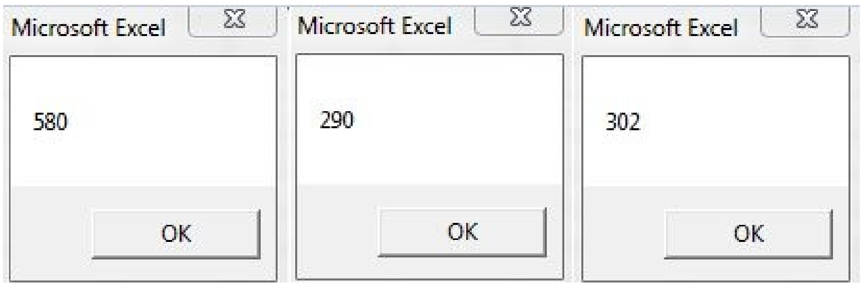
OUTPUT

13.) How to generate a Random Number?



```
Sub Generate_Random()
'This code generates a random number.
    RandomNumber = Int((1000 - 1 + 1) * Rnd + 1)
    MsgBox RandomNumber
End Sub
```

Output is as follows –



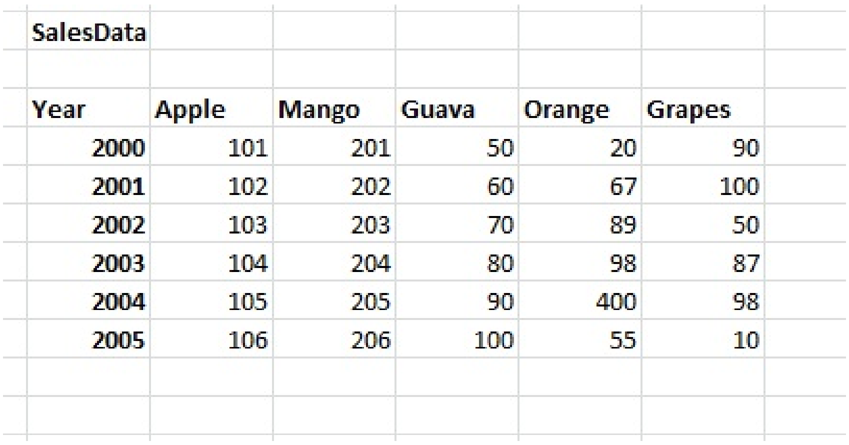
```
Microsoft Excel
580
OK
```

```
Microsoft Excel
290
OK
```

```
Microsoft Excel
302
OK
```

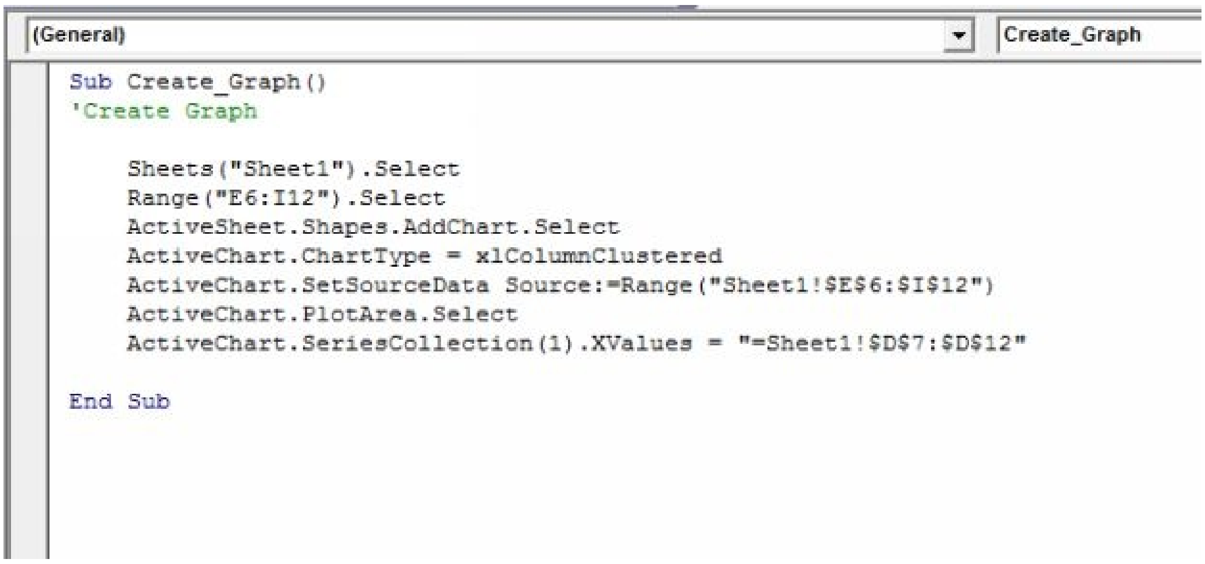
14.) Creating Graph Using VBA. See the code below –

Let’s Say we have the following data. We will make a code to represent this with a graph.



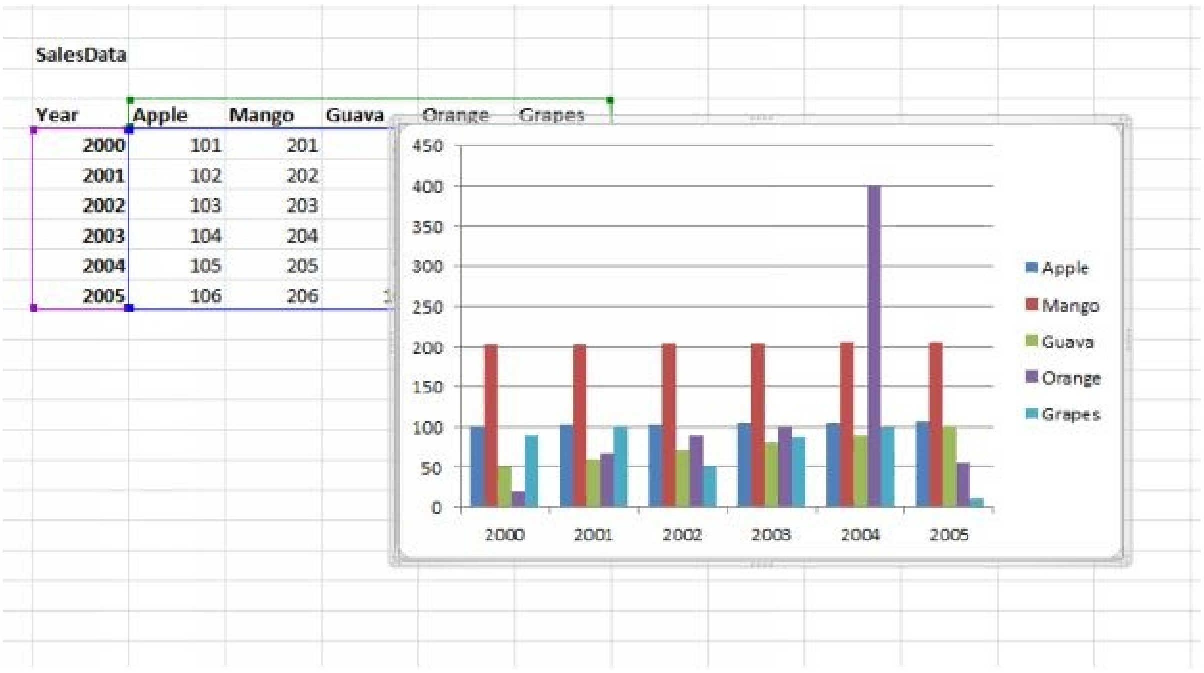
SalesData					
Year	Apple	Mango	Guava	Orange	Grapes
2000	101	201	50	20	90
2001	102	202	60	67	100
2002	103	203	70	89	50
2003	104	204	80	98	87
2004	105	205	90	400	98
2005	106	206	100	55	10

Code- By changing the desired values in the code below you can create a graph for any data.



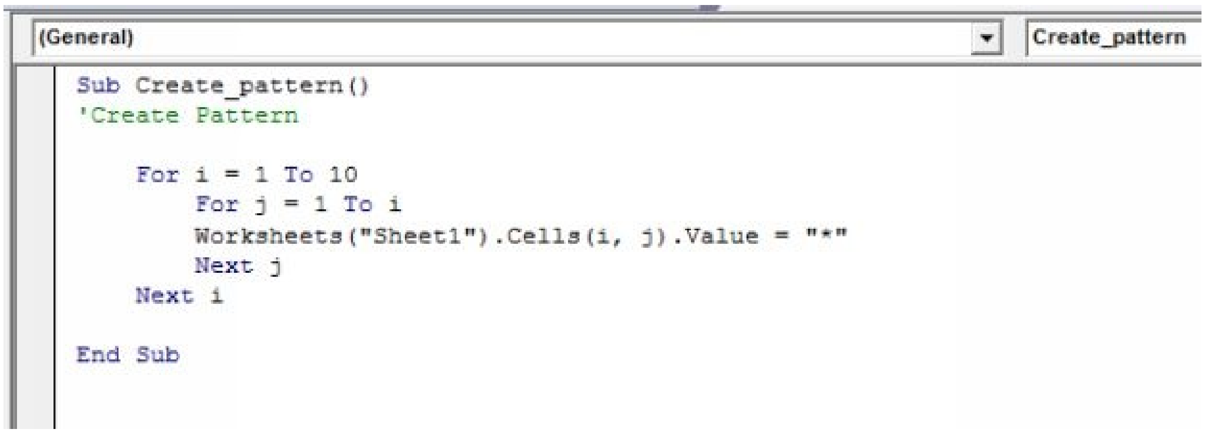
```
Sub Create_Graph()
'Create Graph

    Sheets("Sheet1").Select
    Range("E6:I12").Select
    ActiveSheet.Shapes.AddChart.Select
    ActiveChart.ChartType = xlColumnClustered
    ActiveChart.SetSourceData Source:=Range("Sheet1!$E$6:$I$12")
    ActiveChart.PlotArea.Select
    ActiveChart.SeriesCollection(1).XValues = "=Sheet1!$D$7:$D$12"

End Sub
```

Output –



SalesData					
Year	Apple	Mango	Guava	Orange	Grapes
2000	101	201			
2001	102	202			
2002	103	203			
2003	104	204			
2004	105	205			
2005	106	206	1		

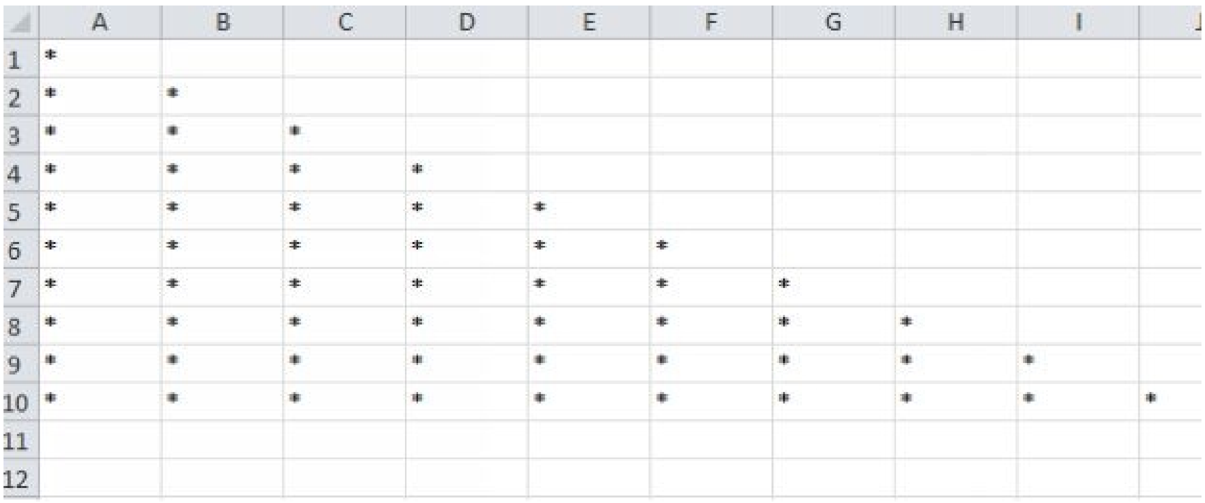
15.) Create a pattern using nested loops – The code below will teach you how to use nested loops and creating patterns of various kinds by just manipulating these loops.



(General) Create_pattern

```
Sub Create_pattern()
'Create Pattern

    For i = 1 To 10
        For j = 1 To i
        Worksheets("Sheet1").Cells(i, j).Value = "*"
        Next j
    Next i

End Sub
```

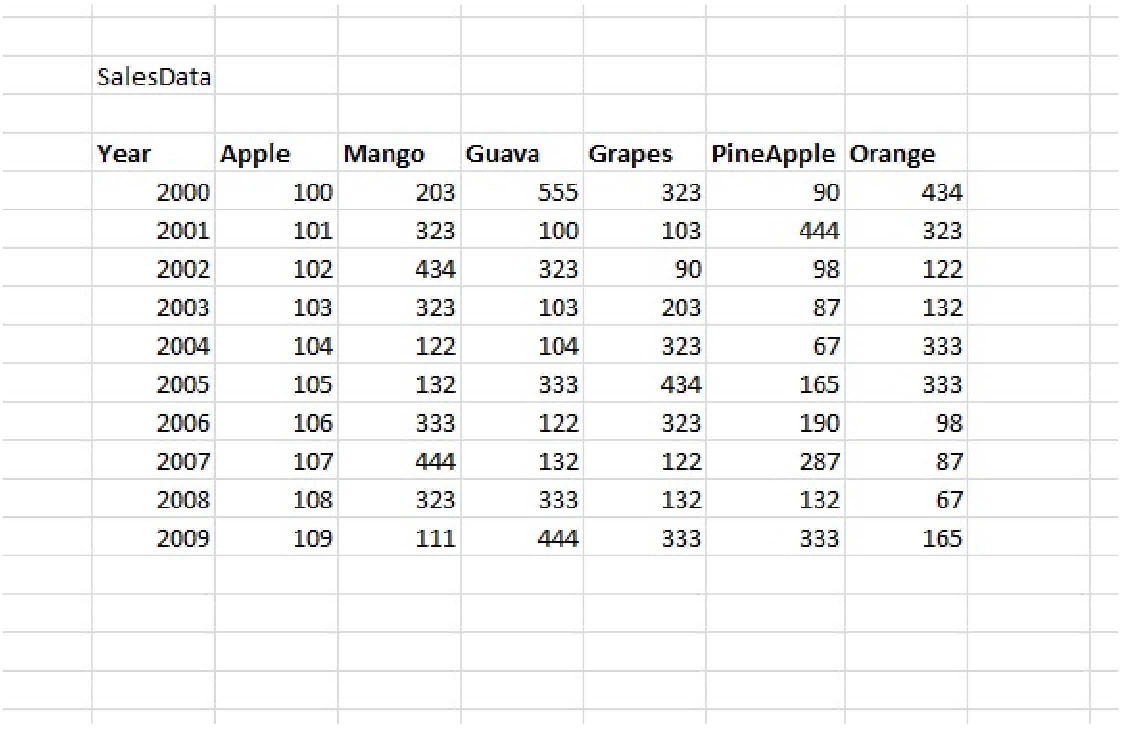
Output –



	A	B	C	D	E	F	G	H	I	J
1	*									
2	*	*								
3	*	*	*							
4	*	*	*	*						
5	*	*	*	*	*					
6	*	*	*	*	*	*				
7	*	*	*	*	*	*	*			
8	*	*	*	*	*	*	*	*		
9	*	*	*	*	*	*	*	*	*	
10	*	*	*	*	*	*	*	*	*	*
11										
12										

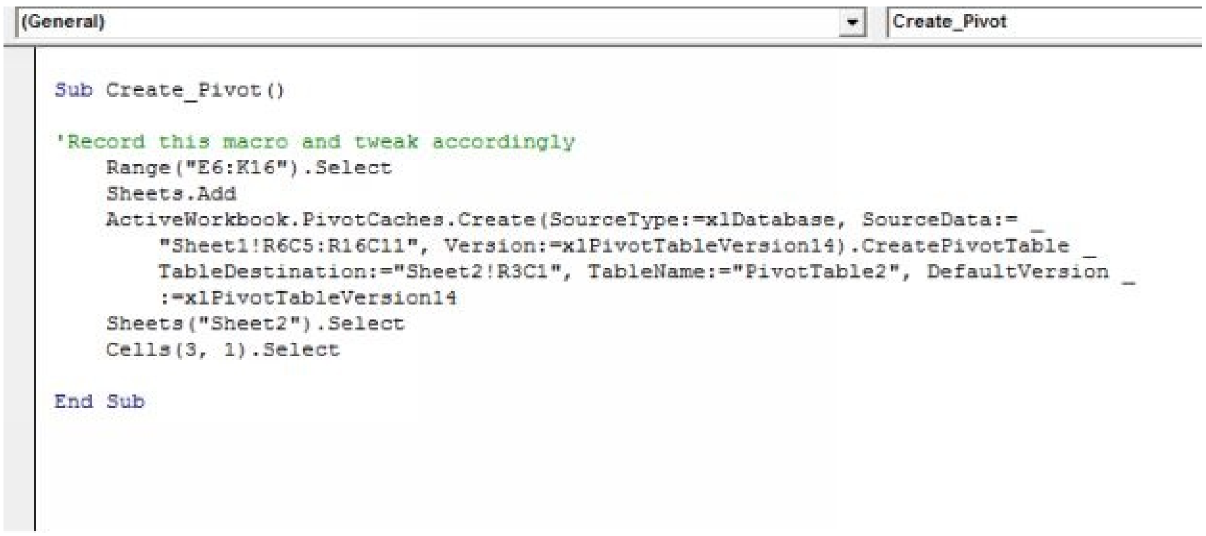
16.) Creating pivot by recording and then tweak it as per your requirement.

Let’s take the same old data –


SalesData

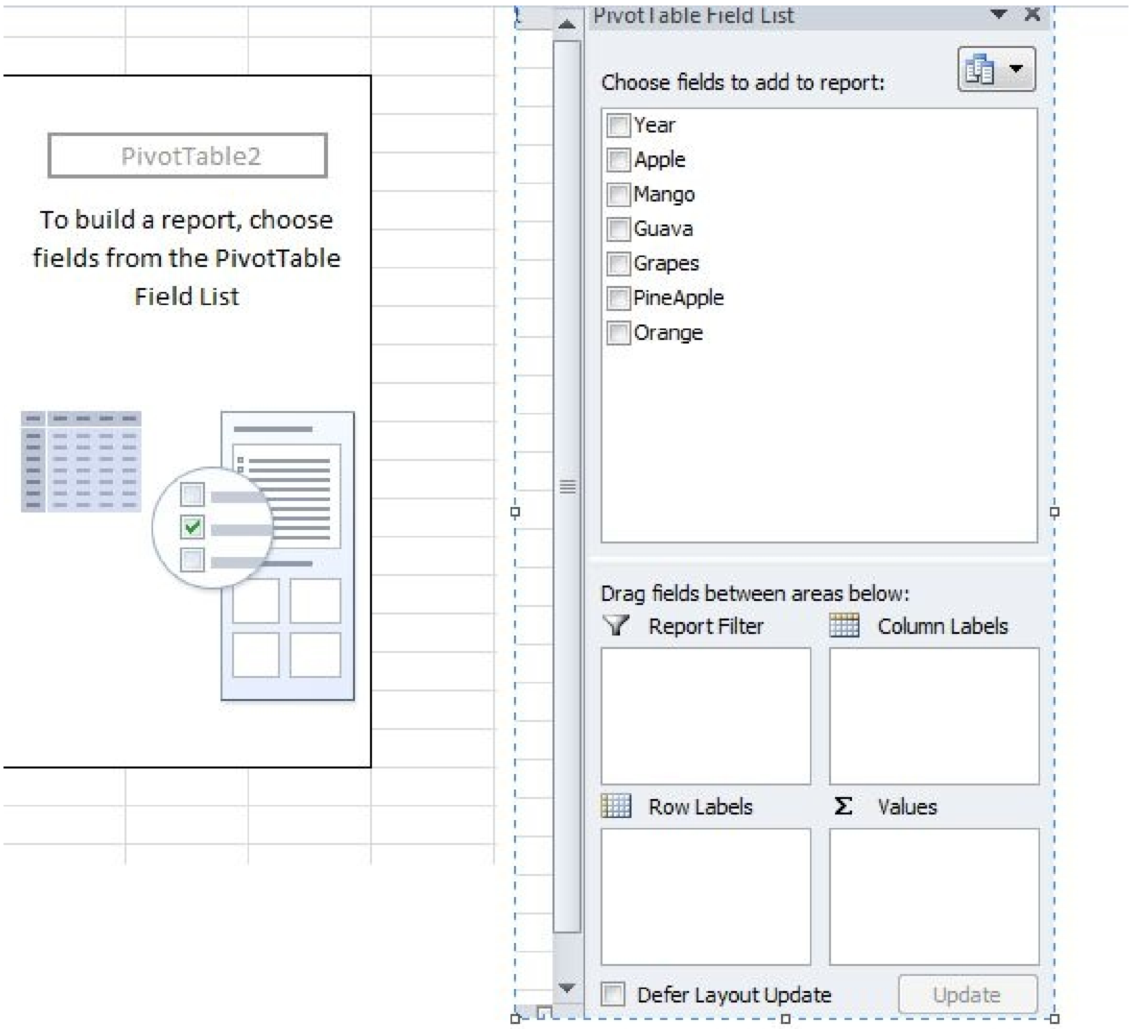
Year	Apple	Mango	Guava	Grapes	PineApple	Orange
2000	100	203	555	323	90	434
2001	101	323	100	103	444	323
2002	102	434	323	90	98	122
2003	103	323	103	203	87	132
2004	104	122	104	323	67	333
2005	105	132	333	434	165	333
2006	106	333	122	323	190	98
2007	107	444	132	122	287	87
2008	108	323	333	132	132	67
2009	109	111	444	333	333	165

Recorded code in VBA –



```
Sub Create_Pivot()

'Record this macro and tweak accordingly
    Range("E6:K16").Select
    Sheets.Add
    ActiveWorkbook.PivotCaches.Create(SourceType:=xlDatabase, SourceData:= _
        "Sheet1!R6C5:R16C11", Version:=xlPivotTableVersion14).CreatePivotTable _
        TableDestination:="Sheet2!R3C1", TableName:="PivotTable2", DefaultVersion _
        :=xlPivotTableVersion14
    Sheets("Sheet2").Select
    Cells(3, 1).Select

End Sub
```

Output –


PivotTable2

To build a report, choose fields from the PivotTable Field List

PivotTable Field List

Choose fields to add to report:

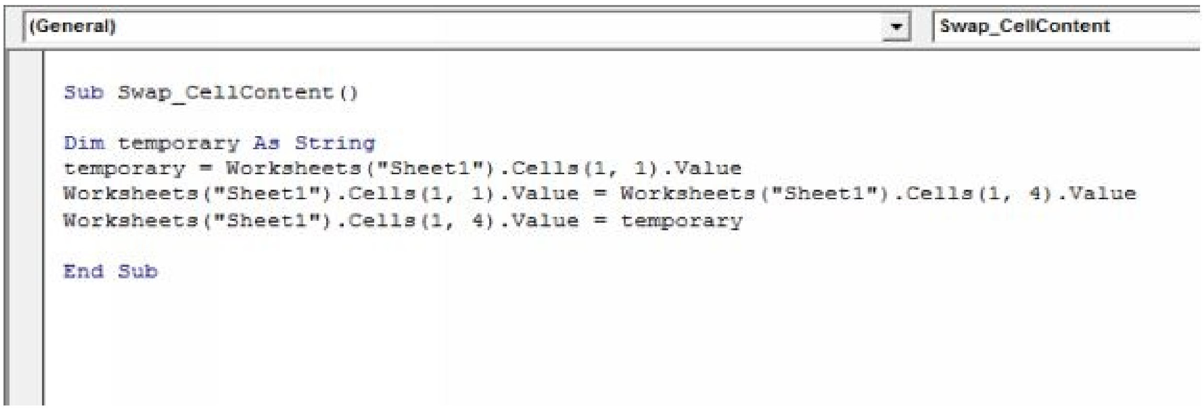
- Year
- Apple
- Mango
- Guava
- Grapes
- PineApple
- Orange

Drag fields between areas below:

Report Filter | Column Labels | Row Labels | Values

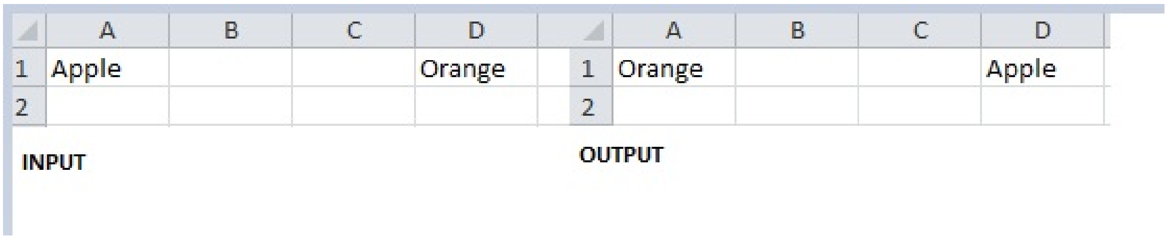
Defer Layout Update | Update

17.) Code to swap values present in cells.


(General) | Swap_CellContent

```
Sub Swap_CellContent()

Dim temporary As String
temporary = Worksheets("Sheet1").Cells(1, 1).Value
Worksheets("Sheet1").Cells(1, 1).Value = Worksheets("Sheet1").Cells(1, 4).Value
Worksheets("Sheet1").Cells(1, 4).Value = temporary

End Sub
```

Input and Output after code execution –

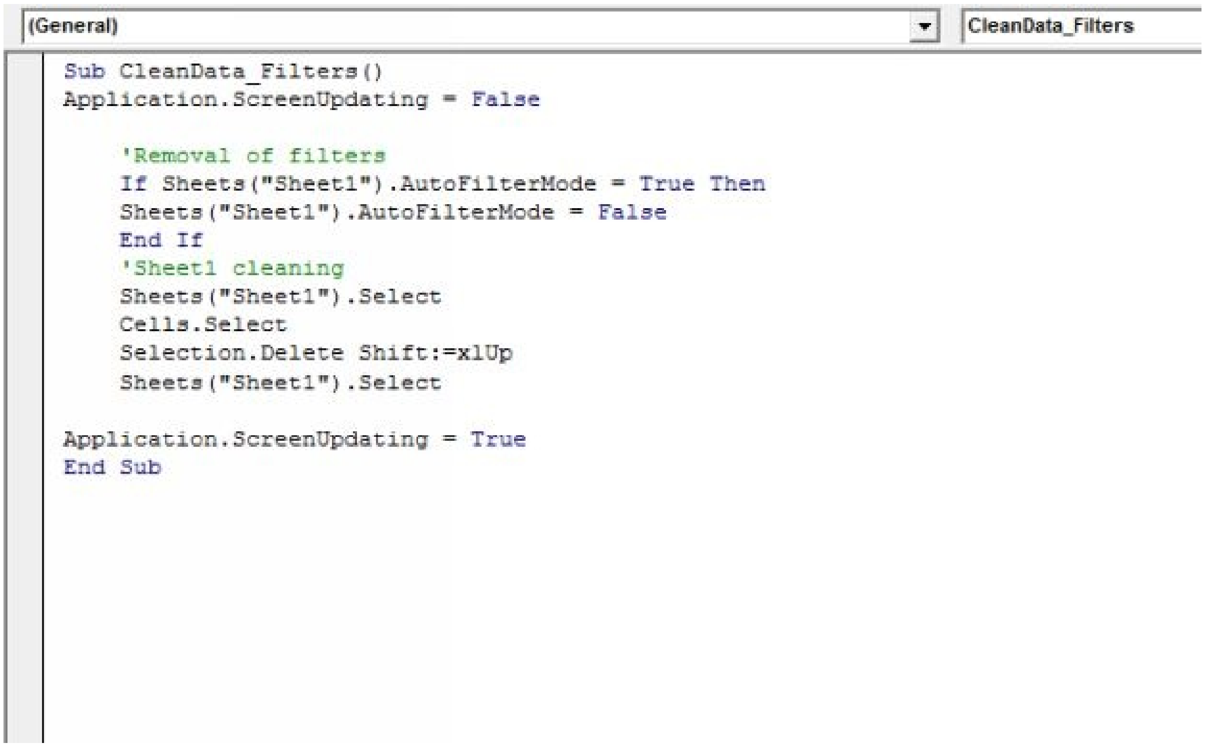

INPUT

	A	B	C	D
1	Apple			Orange
2				

OUTPUT

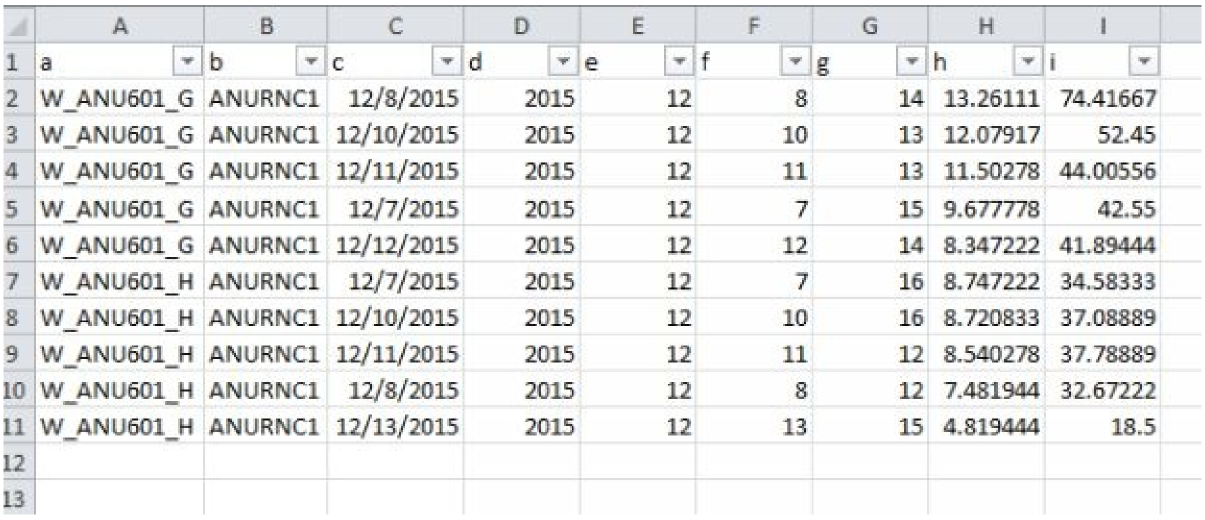
	A	B	C	D
1	Orange			Apple
2				

18.) VBA code to delete all data from the sheet and removing the filters if they exist.



```
Sub CleanData_Filters()
Application.ScreenUpdating = False

    'Removal of filters
    If Sheets("Sheet1").AutoFilterMode = True Then
    Sheets("Sheet1").AutoFilterMode = False
    End If
    'Sheet1 cleaning
    Sheets("Sheet1").Select
    Cells.Select
    Selection.Delete Shift:=xlUp
    Sheets("Sheet1").Select

Application.ScreenUpdating = True
End Sub
```

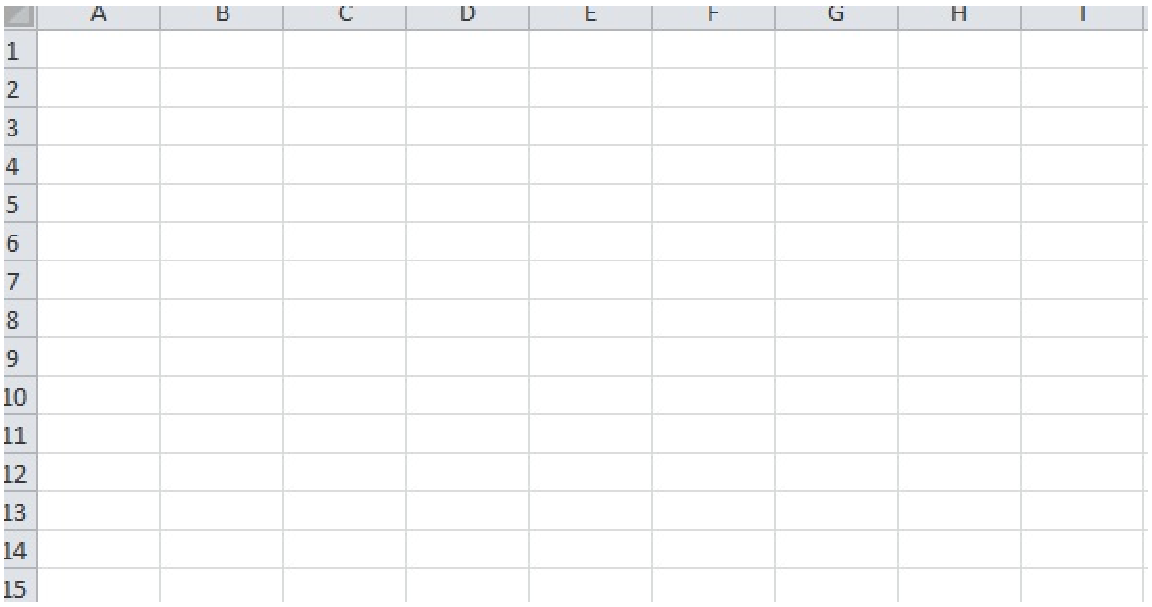
Input Sheet with Data and filters applied



	A	B	C	D	E	F	G	H	I
1	a	b	c	d	e	f	g	h	i
2	W_ANU601_G	ANURNC1	12/8/2015	2015	12	8	14	13.26111	74.41667
3	W_ANU601_G	ANURNC1	12/10/2015	2015	12	10	13	12.07917	52.45
4	W_ANU601_G	ANURNC1	12/11/2015	2015	12	11	13	11.50278	44.00556
5	W_ANU601_G	ANURNC1	12/7/2015	2015	12	7	15	9.677778	42.55
6	W_ANU601_G	ANURNC1	12/12/2015	2015	12	12	14	8.347222	41.89444
7	W_ANU601_H	ANURNC1	12/7/2015	2015	12	7	16	8.747222	34.58333
8	W_ANU601_H	ANURNC1	12/10/2015	2015	12	10	16	8.720833	37.08889
9	W_ANU601_H	ANURNC1	12/11/2015	2015	12	11	12	8.540278	37.78889
10	W_ANU601_H	ANURNC1	12/8/2015	2015	12	8	12	7.481944	32.67222
11	W_ANU601_H	ANURNC1	12/13/2015	2015	12	13	15	4.819444	18.5
12									
13									

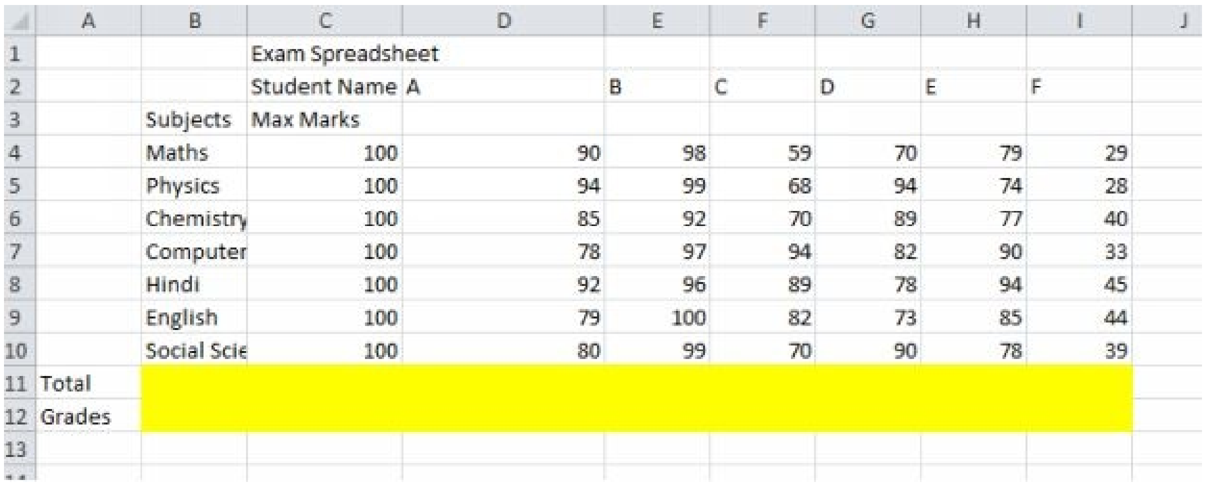
Output after the code is executed –

All the data has been deleted and filters have been removed.



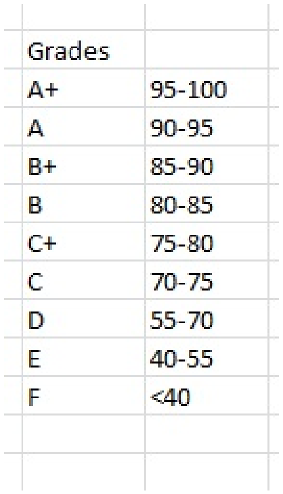
19.) VBA code to create an Exam spreadsheet of students and their grade submissions.

Input Sheet containing student names and marks obtained in various subjects.



	A	B	C	D	E	F	G	H	I	J
1			Exam Spreadsheet							
2			Student Name	A	B	C	D	E	F	
3		Subjects	Max Marks							
4		Maths	100	90	98	59	70	79	29	
5		Physics	100	94	99	68	94	74	28	
6		Chemistry	100	85	92	70	89	77	40	
7		Computer	100	78	97	94	82	90	33	
8		Hindi	100	92	96	89	78	94	45	
9		English	100	79	100	82	73	85	44	
10		Social Scie	100	80	99	70	90	78	39	
11	Total									
12	Grades									
13										

Grade Criteria –



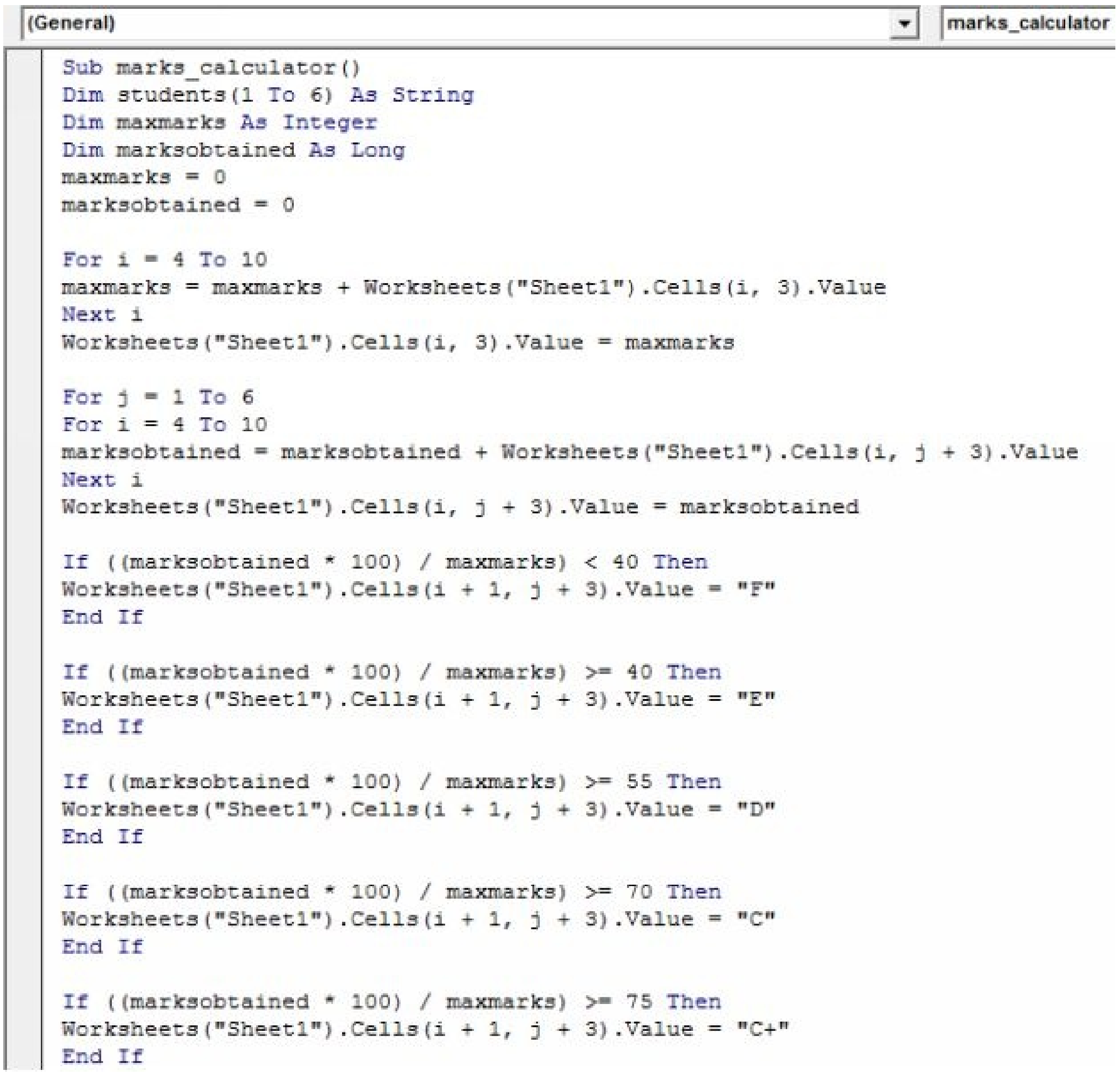
Grades	
A+	95-100
A	90-95
B+	85-90
B	80-85
C+	75-80
C	70-75
D	55-70
E	40-55
F	<40

VBA Code for calculations –

```
Sub marks_calculator ()
Dim students(1 To 6) As String
Dim maxmarks As Integer
Dim marksobtained As Long
maxmarks = 0
marksobtained = 0
For i = 4 To 10
maxmarks = maxmarks + Worksheets(“Sheet1”).Cells(i, 3).Value
Next i
Worksheets(“Sheet1”).Cells(i, 3).Value = maxmarks
For j = 1 To 6
For i = 4 To 10
marksobtained = marksobtained + Worksheets(“Sheet1”).Cells(i, j + 3).Value
Next i
Worksheets(“Sheet1”).Cells(i, j + 3).Value = marksobtained
If ((marksobtained * 100) / maxmarks) < 40 Then
Worksheets(“Sheet1”).Cells(i + 1, j + 3).Value = “F”
End If
If ((marksobtained * 100) / maxmarks) >= 40 Then
Worksheets(“Sheet1”).Cells(i + 1, j + 3).Value = “E”
End If
If ((marksobtained * 100) / maxmarks) >= 55 Then
Worksheets(“Sheet1”).Cells(i + 1, j + 3).Value = “D”
End If
If ((marksobtained * 100) / maxmarks) >= 70 Then
Worksheets(“Sheet1”).Cells(i + 1, j + 3).Value = “C”
End If
If ((marksobtained * 100) / maxmarks) >= 75 Then
Worksheets(“Sheet1”).Cells(i + 1, j + 3).Value = “C+”
End If
If ((marksobtained * 100) / maxmarks) >= 80 Then
Worksheets(“Sheet1”).Cells(i + 1, j + 3).Value = “B”
```

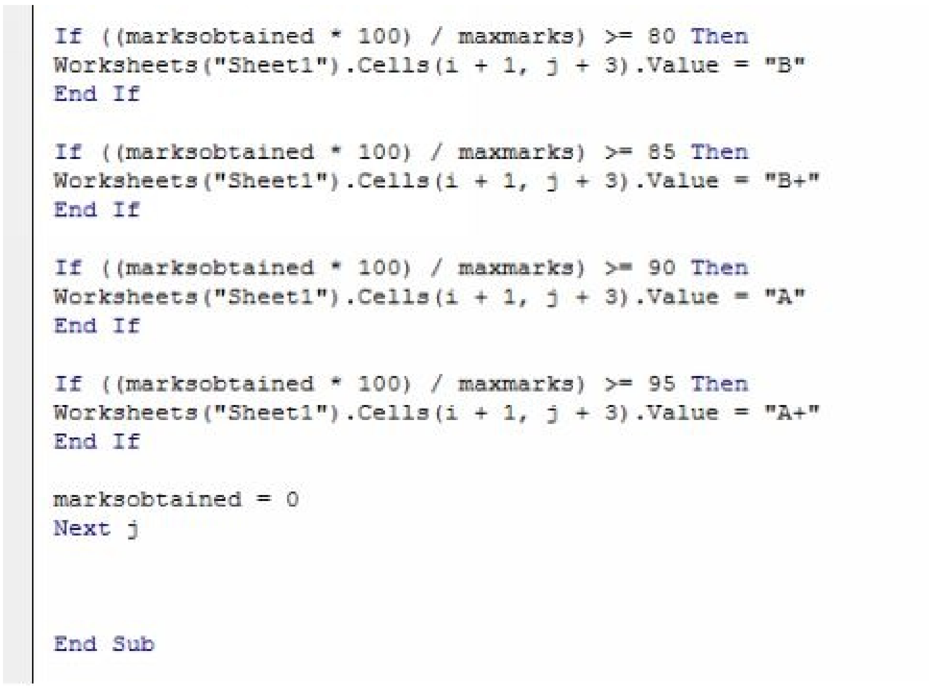
```
End If
If ((marksobtained * 100) / maxmarks) >= 85 Then
Worksheets(“Sheet1”).Cells(i + 1, j + 3).Value = “B+”
End If
If ((marksobtained * 100) / maxmarks) >= 90 Then
Worksheets(“Sheet1”).Cells(i + 1, j + 3).Value = “A”
End If
If ((marksobtained * 100) / maxmarks) >= 95 Then
Worksheets(“Sheet1”).Cells(i + 1, j + 3).Value = “A+”
End If
marksobtained = 0
Next j
End Sub
```

Snapshot of code in IDE



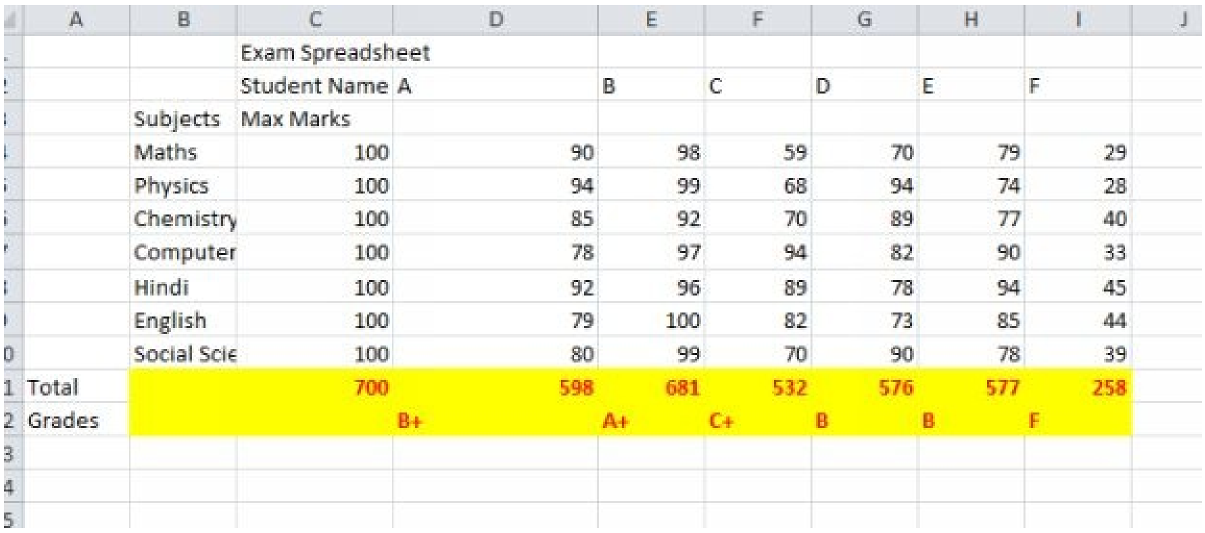
(General) marks_calculator

```
Sub marks_calculator()
Dim students(1 To 6) As String
Dim maxmarks As Integer
Dim marksobtained As Long
maxmarks = 0
marksobtained = 0

For i = 4 To 10
maxmarks = maxmarks + Worksheets("Sheet1").Cells(i, 3).Value
Next i
Worksheets("Sheet1").Cells(i, 3).Value = maxmarks

For j = 1 To 6
For i = 4 To 10
marksobtained = marksobtained + Worksheets("Sheet1").Cells(i, j + 3).Value
Next i
Worksheets("Sheet1").Cells(i, j + 3).Value = marksobtained

If ((marksobtained * 100) / maxmarks) < 40 Then
Worksheets("Sheet1").Cells(i + 1, j + 3).Value = "F"
End If

If ((marksobtained * 100) / maxmarks) >= 40 Then
Worksheets("Sheet1").Cells(i + 1, j + 3).Value = "E"
End If

If ((marksobtained * 100) / maxmarks) >= 55 Then
Worksheets("Sheet1").Cells(i + 1, j + 3).Value = "D"
End If

If ((marksobtained * 100) / maxmarks) >= 70 Then
Worksheets("Sheet1").Cells(i + 1, j + 3).Value = "C"
End If

If ((marksobtained * 100) / maxmarks) >= 75 Then
Worksheets("Sheet1").Cells(i + 1, j + 3).Value = "C+"
End If
```



```
If ((marksobtained * 100) / maxmarks) >= 80 Then
Worksheets("Sheet1").Cells(i + 1, j + 3).Value = "B"
End If

If ((marksobtained * 100) / maxmarks) >= 85 Then
Worksheets("Sheet1").Cells(i + 1, j + 3).Value = "B+"
End If

If ((marksobtained * 100) / maxmarks) >= 90 Then
Worksheets("Sheet1").Cells(i + 1, j + 3).Value = "A"
End If

If ((marksobtained * 100) / maxmarks) >= 95 Then
Worksheets("Sheet1").Cells(i + 1, j + 3).Value = "A+"
End If

marksobtained = 0
Next j



End Sub
```

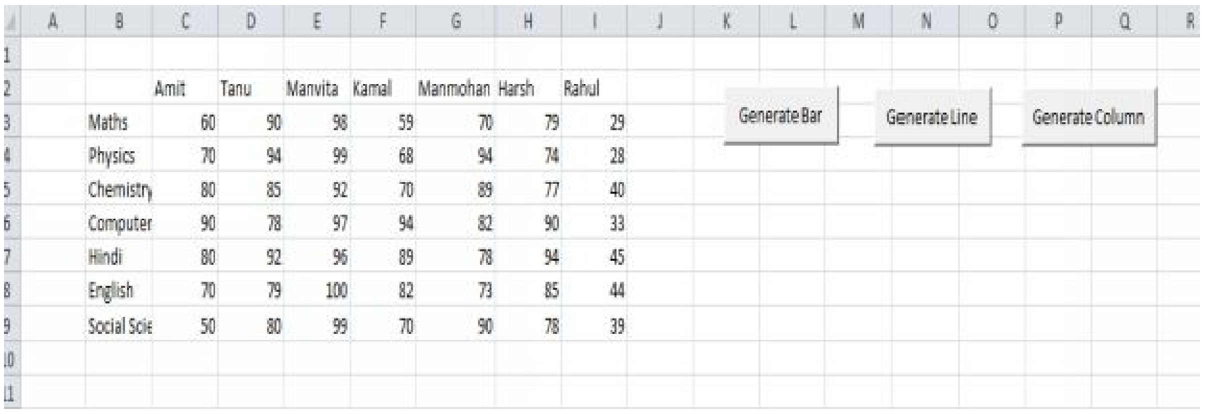
See Output in Bold Red below –



	A	B	C	D	E	F	G	H	I	J
1			Exam Spreadsheet							
2			Student Name	A	B	C	D	E	F	
3		Subjects	Max Marks							
4		Maths	100	90	98	59	70	79	29	
5		Physics	100	94	99	68	94	74	28	
6		Chemistry	100	85	92	70	89	77	40	
7		Computer	100	78	97	94	82	90	33	
8		Hindi	100	92	96	89	78	94	45	
9		English	100	79	100	82	73	85	44	
10		Social Scie	100	80	99	70	90	78	39	
11	Total		700	598	681	532	576	577	258	
12	Grades			B+	A+	C+	B	B	F	
13										
14										
15										

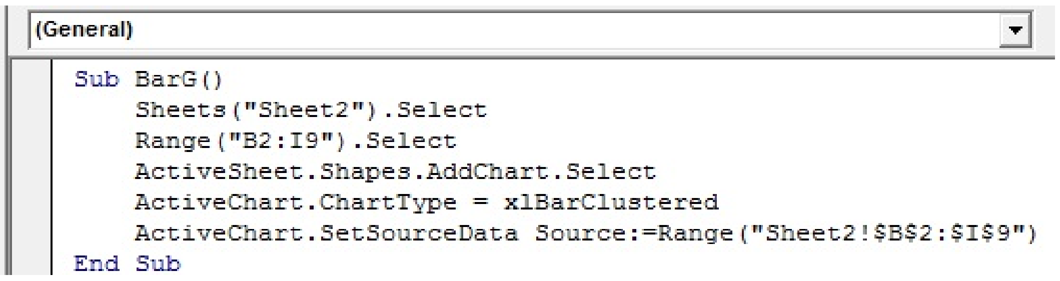
20.) Generating different types of graph from a given data.

Let’s see the data first. Look Below for the data and the control buttons which will show different types of graphs on clicking.



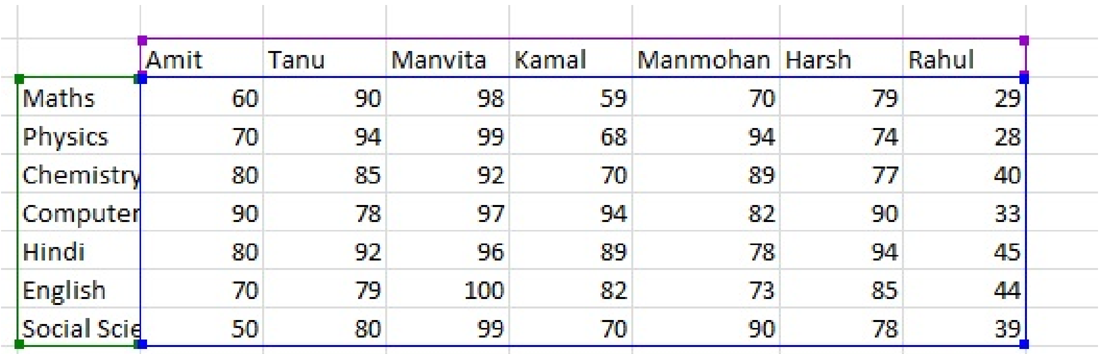
	A	B	C	D	E	F	G	H	I
1									
2			Amit	Tanu	Manvita	Kamal	Manmohan	Harsh	Rahul
3		Maths	60	90	98	59	70	79	29
4		Physics	70	94	99	68	94	74	28
5		Chemistry	80	85	92	70	89	77	40
6		Computer	90	78	97	94	82	90	33
7		Hindi	80	92	96	89	78	94	45
8		English	70	79	100	82	73	85	44
9		Social Scie	50	80	99	70	90	78	39
10									
11									

Code for Generating Bar Graph is as follows –

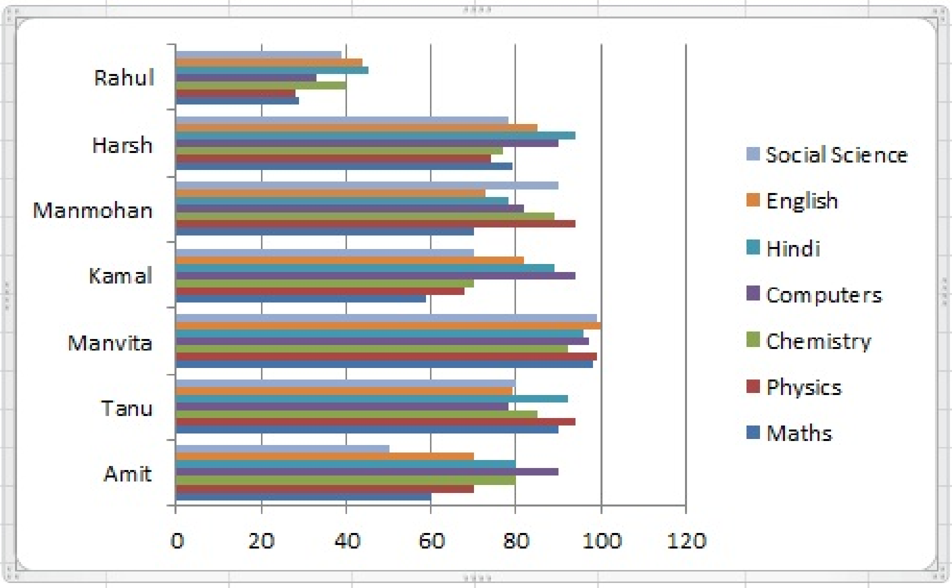


```
Sub BarG()
    Sheets("Sheet2").Select
    Range("B2:I9").Select
    ActiveSheet.Shapes.AddChart.Select
    ActiveChart.ChartType = xlBarClustered
    ActiveChart.SetSourceData Source:=Range("Sheet2!$B$2:$I$9")
End Sub
```

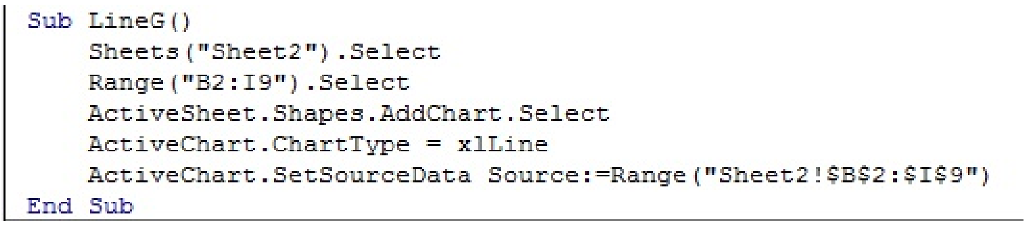
Output –



	Amit	Tanu	Manvita	Kamal	Manmohan	Harsh	Rahul
Maths	60	90	98	59	70	79	29
Physics	70	94	99	68	94	74	28
Chemistry	80	85	92	70	89	77	40
Computer	90	78	97	94	82	90	33
Hindi	80	92	96	89	78	94	45
English	70	79	100	82	73	85	44
Social Scie	50	80	99	70	90	78	39

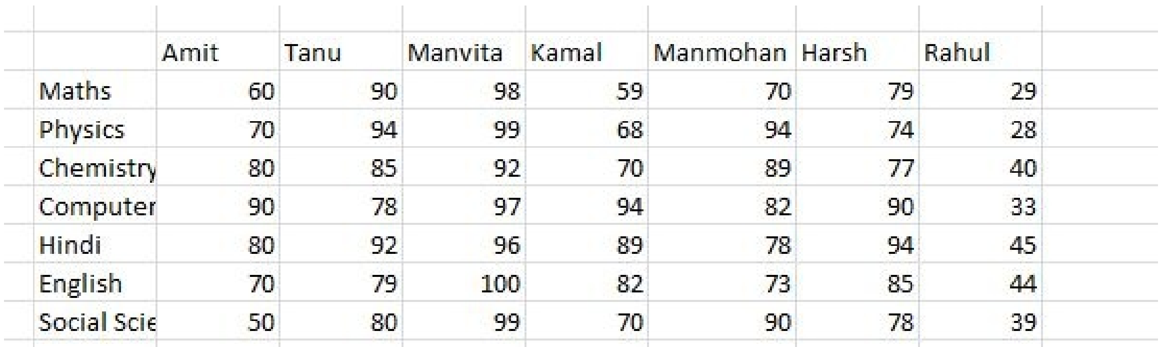


Code for Generating Line Graph is as follows –

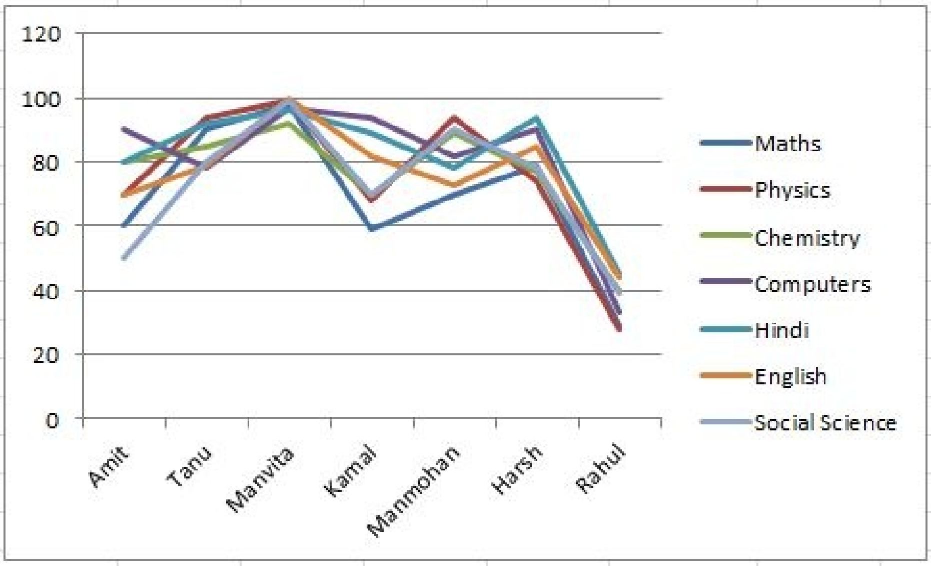


```
Sub LineG()
    Sheets("Sheet2").Select
    Range("B2:I9").Select
    ActiveSheet.Shapes.AddChart.Select
    ActiveChart.ChartType = xlLine
    ActiveChart.SetSourceData Source:=Range("Sheet2!$B$2:$I$9")
End Sub
```

Output is as follows –



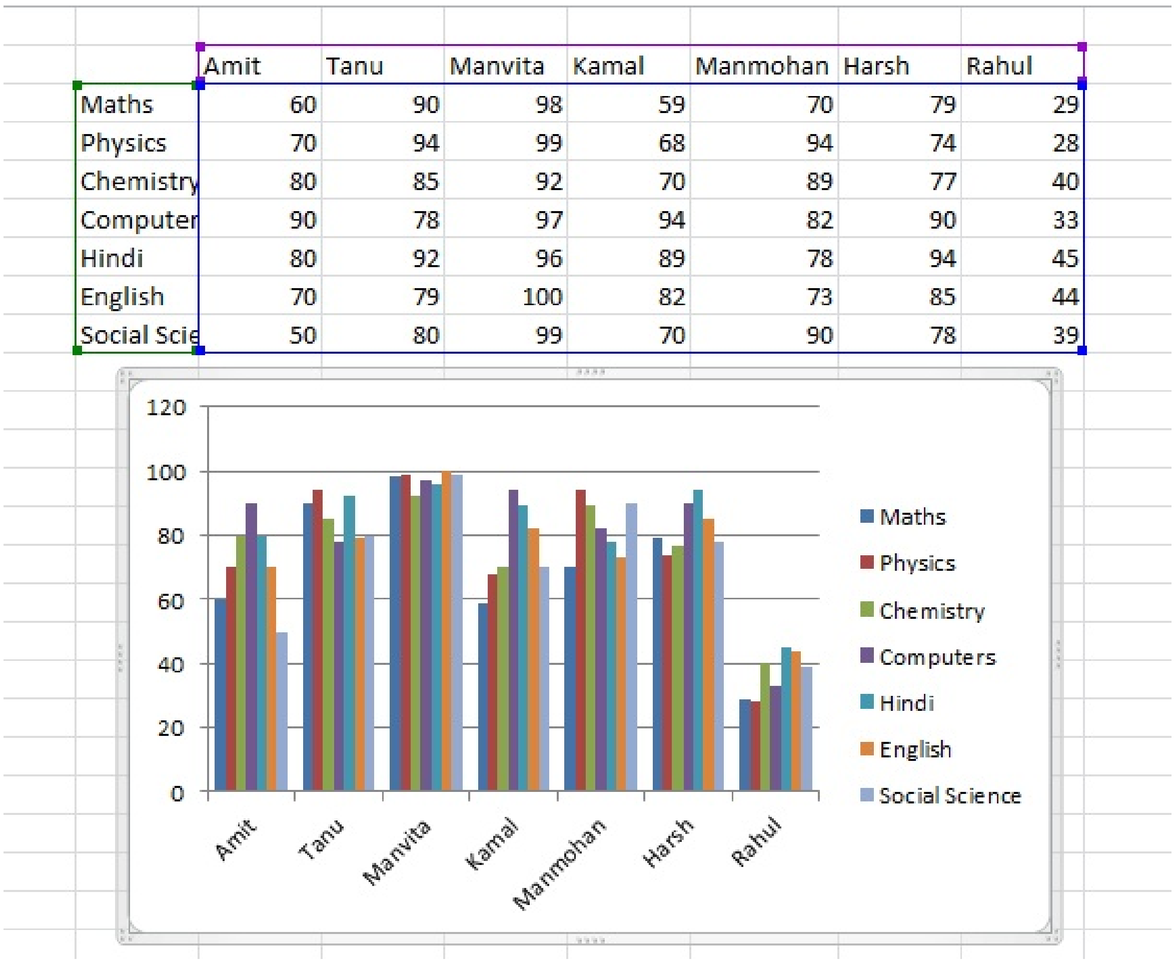
	Amit	Tanu	Manvita	Kamal	Manmohan	Harsh	Rahul
Maths	60	90	98	59	70	79	29
Physics	70	94	99	68	94	74	28
Chemistry	80	85	92	70	89	77	40
Computer	90	78	97	94	82	90	33
Hindi	80	92	96	89	78	94	45
English	70	79	100	82	73	85	44
Social Scie	50	80	99	70	90	78	39



Code for Chart Type of Graph –

```
Sub ChartG()
 Sheets("Sheet2").Select
    Range("B2:I9").Select
    ActiveSheet.Shapes.AddChart.Select
    ActiveChart.ChartType = xlColumnClustered
    ActiveChart.SetSourceData Source:=Range("Sheet2!$B$2:$I$9")
End Sub
```

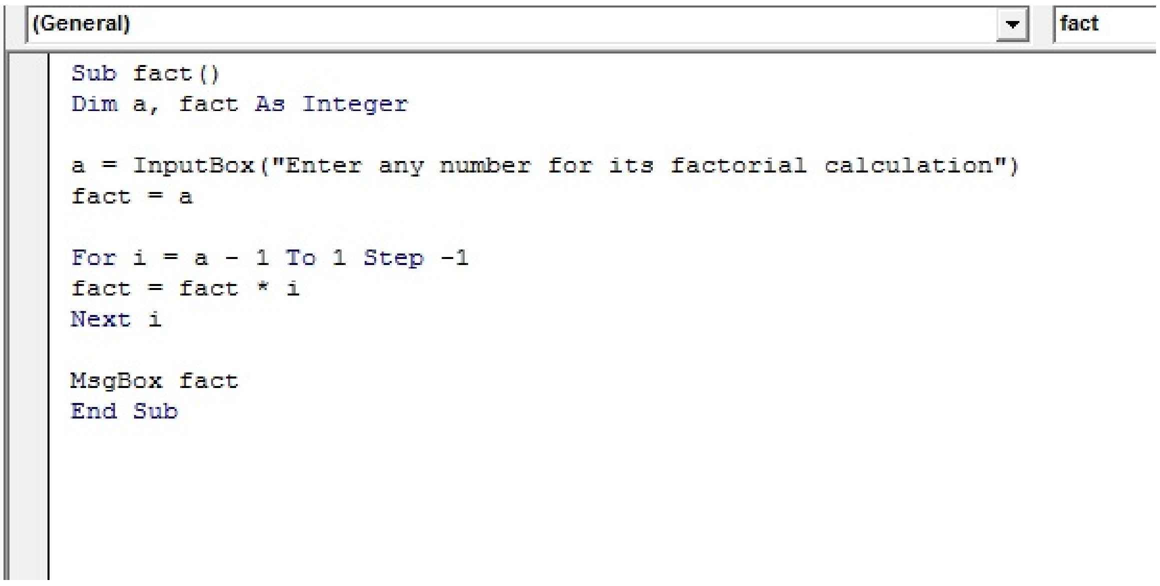
Output is as follows –



	Amit	Tanu	Manvita	Kamal	Manmohan	Harsh	Rahul
Maths	60	90	98	59	70	79	29
Physics	70	94	99	68	94	74	28
Chemistry	80	85	92	70	89	77	40
Computer	90	78	97	94	82	90	33
Hindi	80	92	96	89	78	94	45
English	70	79	100	82	73	85	44
Social Scie	50	80	99	70	90	78	39

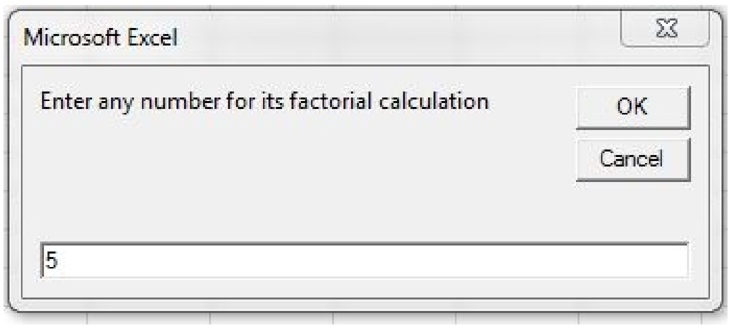
21.) Code to display factorial of a number.

```
Sub fact()
Dim a, fact As Integer
a = InputBox("Enter any number for its factorial calculation")
fact = a
For i = a - 1 To 1 Step -1
fact = fact * i
Next i
MsgBox fact
End Sub
```

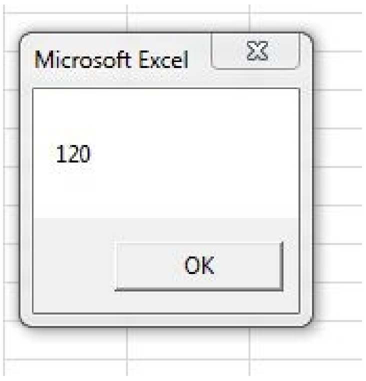


```
Sub fact()
Dim a, fact As Integer

a = InputBox("Enter any number for its factorial calculation")
fact = a

For i = a - 1 To 1 Step -1
fact = fact * i
Next i

MsgBox fact
End Sub
```

Output –



```
Enter any number for its factorial calculation
5
```



```
120
```