Excellence with Excel – Certification Quiz Questions

Module 6 – VBA and Macros

1. **You are writing VBA code to create an "Input Box" macro that will format cells that require user input with a yellow fill color, grey borders, and a blue font color. You want to execute this macro over large ranges to format many cells at once. You've completed the first step of recording the macro and simplifying some of the code, and your subroutine so far is shown below:**

```
Sub Input_Box()
'
' Input_Box Macro
'
' Keyboard Shortcut: Ctrl+Shift+I
'
    With Selection

        .Borders.LineStyle = xlContinuous
        .Borders.Color = -5066062
        .Borders.Weight = xlThin

        .Interior.Pattern = xlSolid
        .Interior.PatternColorIndex = xlAutomatic
        .Interior.Color = 10092543

        .Font.Color = -65536

        .HorizontalAlignment = xlCenter
    End With
End Sub
```

Which of the following answer choices represent(s) LIMITATIONS or PROBLEMS with this VBA code that you should fix to make a more robust macro?

a. This macro will not work across a range of cells – only with the one specific active cell on the spreadsheet.

b. This macro will apply the formatting even to a blank cell, which is not correct since blank cells should not be input boxes.

c. If the selected cell is far beyond the UsedRange of the spreadsheet, this macro will still apply the formatting to it – which is not correct.

d. This macro applies the same font color to all types of cells, so it does not color-code constants in blue and formulas in black.

e. Answer choices 1, 2, 3, and 4.

f. Answer choices 2, 3, and 4.

g. Answer choices 1, 2, and 4.

h. Answer choices 1, 3, and 4.

2. You've now enhanced the macro above to fix some of these issues. Your current code for this subroutine is shown below:

```
Sub Input_Box()
'
' Input_Box Macro
'
' Keyboard Shortcut: Ctrl+Shift+I
'
    Dim NonEmptyCells As Range, FormulaCells As Range, ConstantCells As Range

    On Error Resume Next

    FormulaCells = Intersect(Selection, Selection.SpecialCells(xlCellTypeFormulas))
    ConstantCells = Intersect(Selection, Selection.SpecialCells(xlCellTypeConstants, xlNumbers))

    NonEmptyCells = Union(FormulaCells, ConstantCells)
    FormulaCells.Font.Color = RGB(0, 0, 0)
    ConstantCells.Font.Color = RGB(0, 0, 255)

    If Not NonEmptyCells Is Nothing Then
        With NonEmptyCells
            .Borders.LineStyle = xlContinuous
            .Borders.Color = RGB(178, 178, 178)
            .Borders.Weight = xlThin

            .Interior.Pattern = xlSolid
            .Interior.PatternColorIndex = xlAutomatic
            .Interior.Color = RGB(255, 255, 153)

            .HorizontalAlignment = xlCenter
        End With
    End If

    On Error GoTo 0

End Sub
```

However, this version of the macro is not working correctly. What is the problem?

a. The Intersect functions are incorrect – you should add ActiveSheet.UsedRange to the input variables in addition to the two already there.

b. The "On Error Resume Next" command is unnecessary and is causing the subroutine to exit, even when there's no problem.

c. You've failed to use the "Set" command in front of the Range variables when setting them equal to intersections and unions of other ranges.

d. You cannot combine two ranges of cells with the Union function because it works only if the ranges have at least one cell of overlap.

e. If FormulaCells or ConstantCells are empty, then the Union function will not work correctly, nor will the .Font.Color assignments below it.

f. Answer choices 1, 2, 3, 4, and 5.

g. Answer choices 1, 2, 3, and 4.

h. Answer choices 1, 3, 4, and 5.

i. Answer choices 1, 3, and 5.

j. Answer choices 1, 3, and 4.

3. **You're working on a macro to color-code cells in a financial model properly. As part of this process, you've written code that selects constant cells that have text, constant cells that have numbers, formula cells that have text, and formula cells that have numbers. Then, you loop through the individual cells in each range and change the colors appropriately. The portion for the numeric constant cells is shown below:**

```
If Not numConstantCells Is Nothing Then
    For Each cell In numConstantCells
        If cell.Interior.Color = RGB(255, 255, 153) Or cell.Interior.Color = RGB(255, 255, 255) Then
            cell.Font.Color = RGB(0, 0, 255)
        End If
    Next cell
End If
```

Your co-worker reviews this code and says that a "For Each" loop is inefficient because you could use "With numConstantCells" and apply the blue font color to everything in the range at once; there's no need to check the interior colors. Is your co-worker correct?

a. Yes – it's always more efficient to use the "With" and "End With" commands to change properties in ranges.

b. No – you need to loop through each cell in this range to check its interior fill color and make sure it's yellow or white before changing the font color.

c. No – it's not more efficient to use "With" and "End With" rather than looping through each cell in a range and changing the properties manually.

d. Maybe – it depends on the setup of this spreadsheet and the most common interior fill colors for numeric constant cells.

4. **You are writing a VBA subroutine to loop through a selection of cells and increment each cell's value by 1, but ONLY if that cell contains a non-date number AND that cell's value is positive. Which of the following screenshots represents the BEST code to accomplish this (assume that numConstantCells has been set up correctly)?**

a.

```
If Not numConstantCells Is Nothing Then
    For Each cell In numConstantCells
        If IsNumeric(cell.Value) = True Then
            If cell.Value > 0 Then
                cell.Value = cell.Value + 1
            End If
        End If
    Next
End If
```

b.

```
If Not numConstantCells Is Nothing Then
    For Each cell In numConstantCells
        If IsNumeric(cell.Value) = True And cell.Value > 0 Then
            cell.Value = cell.Value + 1
        End If
    Next
End If
```

c.

```
If Not numConstantCells Is Nothing Then
    For Each cell In numConstantCells
        If cell.Value > 0 Then
            cell.Value = cell.Value + 1
        End If
    Next
End If
```

d.

```
If Not numConstantCells Is Nothing Then
    For Each cell In numConstantCells
        If cell.IsNumeric = True And cell.Value > 0 Then
            cell.Value = cell.Value + 1
        End If
    Next
End If
```

5. **You're writing a macro that will loop through a selected range of cells and "cycle" the number formats. In other words, if the cell currently has Number Format 1, it will switch to Number Format 2, and then to Number Format 3. There are 6 formats total, so when the cell is using Number Format 6, the macro will switch it back to Number Format 1.**

   Each number format is stored in an array called "numberFormats", and part of the code is shown below (assume that nonEmptyCells has been filled correctly):

```
If Not nonEmptyCells Is Nothing Then
    For Each cell In nonEmptyCells

        i = 1

        For Each indivFormat In numberFormats

            If StrComp(cell.NumberFormat, indivFormat) = 0 Then
                cell.NumberFormat = numberFormats(i + 1)
                Exit For
            End If

            i = i + 1

            If i = lastFormatNum Then
                cell.NumberFormat = numberFormats(1)
            End If

        Next indivFormat
    Next cell
End If
```

Is there any way to make the code for this part of the macro more efficient?

a. Instead of using StrComp, use the "Like" operator to handle the case where the cell's number format is slightly different from a similar format in the array.

b. In the last If statement, put an "Exit For" right below the cell.NumberFormat = numberFormats(i) line.

c. Instead of checking the number format of every cell in the range, look at only the first cell's number format, advance to the next number format, and apply that same next format to everything else in the range.

d. Not really – no matter what you do, you have to loop through all the cells in the range and then loop through all the number formats for each cell.

6. **You're going to write a macro to shift the number of decimal places for each number in a range of cells up by 1 or down by 1 so that 3.4 becomes 3.40, or 10.63 becomes 10.6. Excel has built-in commands to increase and decrease the decimal places in numbers, but to use these commands, you MUST select the cell first.**

   You plan to loop through each numeric, non-date cell in the range, select it with .Select, and then execute these built-in commands.

Is it a good idea to write the macro this way?

a. No – it's poor practice to select cells in VBA before changing them, as it's very inefficient compared with "With" and "End With" or even looping through the cells to change their properties without selecting anything.

b. Yes – while this method is inefficient, there is no other way to accomplish this in Excel if the numbers are in different formats and have different decimal places.

c. No – it's much better to parse the number formats and replace each ".0" with ".00" and continue doing that for other characters after the decimal place.

d. It's not a great idea to write the macro this way, but the other alternatives are more confusing to write or also have efficiency issues.

7. **You're writing a macro to "flip the signs" of numeric, non-date cells that contain formulas or constants in the selected range of cells. In other words, positive numbers will become negative, and negative numbers will become positive. Part of the code is shown below:**

```
If Not numConstantCells Is Nothing Then
    For Each cell In numConstantCells
        If IsNumeric(cell.Value) = True Then
            cell.Value = -cell.Value
        End If
    Next cell
End If

If Not numFormulaCells Is Nothing Then
    For Each cell In numFormulaCells
        If IsNumeric(cell.Value) = True Then

            firstThreeChars = Left(cell.Formula, 3)

            If StrComp(firstThreeChars, "=-(") = 0 Then
                formulaText = Mid(cell.Formula, 4, Len(cell.Formula) - 4)
                cell.Formula = "=" & formulaText
            Else
                formulaText = Right(cell.Formula, Len(cell.Formula) - 1)
                cell.Formula = "=-(" & formulaText & ")"
            End If

        End If
    Next cell
End If
```

Is it necessary to treat cells with constants and those with formulas differently, as this code does?

a. No – you're doing this only to support the "toggle" feature that lets the user switch from negative to positive and back when activating the macro multiple times.

b. Yes – you must put "-(" and ")" around formulas to make them negative, but you cannot do that with constants.

c. No – but it makes the code more efficient and easier to read, and it's easier to make the macro toggle between positives and negatives like this.

d. Yes – the cell.Formula part will not work if the cell has a constant number rather than a formula.

8. **Which of the following statements describe(s) the CORRECT similarities and differences between User-Defined Functions (UDFs) and Macros in Excel?**

a. You can create and edit both UDFs and Macros in the VBA Editor.

b. Macros are more useful for formatting commands, such as color-coding cells, while UDFs are more useful for making certain calculations that are not built into Excel.

c. You can record the initial versions of both UDFs and Macros with the "recorder" that's built into Excel, and then you can modify them in the VBA Editor.

d. Both Macros and UDFs could start with either "Function" or "Sub" in VBA, depending on whether or not they return a value at the end.

e. Answer choices 1, 2, 3, and 4.

f. Answer choices 1, 2, and 3.

g. Answer choices 1 and 2.

h. Answer choices 1, 3, and 4.