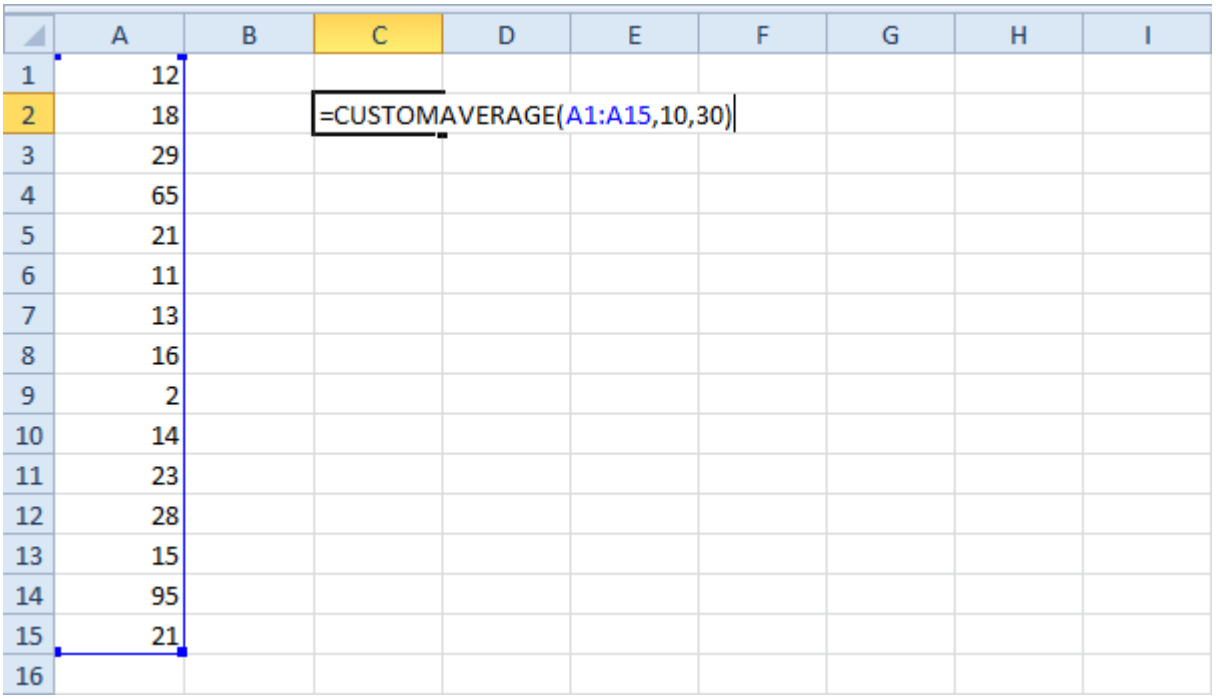
IE 469 INDUSTRIAL APPLICATIONS OF OPERATIONS RESEARCH

SPRING 2024 VBA LAB STUDY 2

Exercise 1: Custom Average

Below we will look at a program in Excel VBA which creates a User Defined Function that calculates the average of a randomly selected range excluding one or more values that are outliers and shouldn't be averaged.



	A	B	C	D	E	F	G	H	I
1	12								
2	18		=CUSTOMAVERAGE(A1:A15,10,30)						
3	29								
4	65								
5	21								
6	11								
7	13								
8	16								
9	2								
10	14								
11	23								
12	28								
13	15								
14	95								
15	21								
16									

User defined functions need to be placed into a module.

1. Open the Visual Basic Editor and click Insert, Module.

2. Add the following code line:

```
Function CUSTOMAVERAGE(rng As Range, lower As Integer, upper As Integer)
```

The name of our Function is CUSTOMAVERAGE. The part between the brackets means we give Excel VBA a range and two Integer variables as input. We name our range rng, one Integer variable we call lower, and one Integer variable we call upper, but you can use any names.

3. Next, we declare a Range object and two variables of type Integer. We call the Range object cell. One Integer variable we call total and one Integer variable we call count.

```
Dim cell As Range, total As Integer, count As Integer
```

4. We want to check each cell in a randomly selected range (this range can be of any size). In Excel VBA, you can use the For Each Next loop for this. Add the following code lines:

```
For Each cell In rng

Next cell
```

Note: rng and cell are randomly chosen here, you can use any names. Remember to refer to these names in the rest of your code.

5. Next, we check for each value in this range if it falls between the two values (lower and upper). If true, we increment total by the value of the cell and we increment count by 1. Add the following code lines to the loop.

```
If cell.Value >= lower And cell.Value <= upper Then
    total = total + cell.Value
    count = count + 1
End If
```

6. To return the result of this function (the desired average), add the following code line outside the loop.

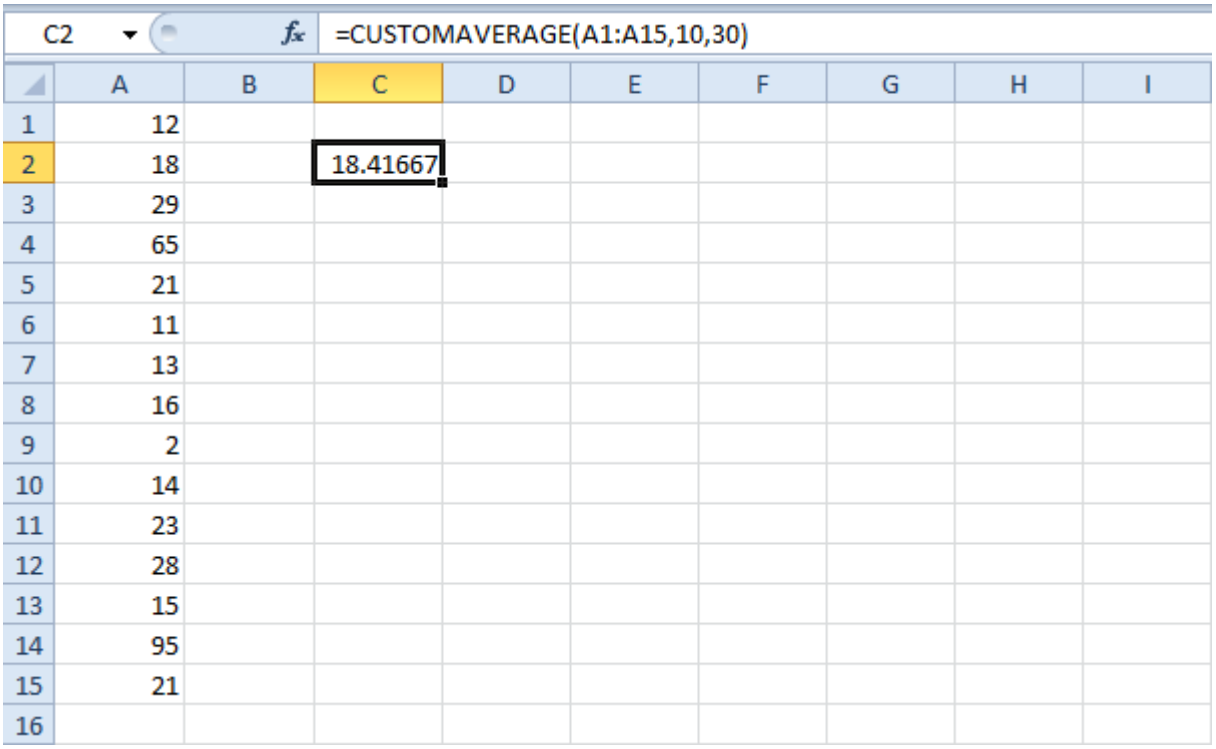
```
CUSTOMAVERAGE = total / count
```

7. Don't forget to end the function. Add the line:

```
End Function
```

8. Now you can use this function just like any other Excel function to calculate the average of numbers that fall between two values.

Result:



	A	B	C	D	E	F	G	H	I
1	12								
2	18		18.41667						
3	29								
4	65								
5	21								
6	11								
7	13								
8	16								
9	2								
10	14								
11	23								
12	28								
13	15								
14	95								
15	21								
16									

As a check, you can delete all values that are lower than 10 and higher than 30 and use the standard Average function in Excel to see if Excel calculates the same average as our custom average function.

C2 =AVERAGE(A1:A15)

	A	B	C	D	E	F	G	H	I
1	12								
2	18		18.41667						
3	29								
4									
5	21								
6	11								
7	13								
8	16								
9									
10	14								
11	23								
12	28								
13	15								
14									
15	21								
16									

Now, modify this code to have the CUSTOMAVERAGE function to have one more parameter in addition to the range, lower and upper. Call this parameter "OddEven" which can be either 0, 1 or 2. (This means that this addition will make the function look like CUSTOMAVERAGE(Range,Min,Max,OddEven). If this parameter is entered as 0, then CUSTOMAVERAGE should work as above (no changes). However, if it is 1 then the CUSTOMAVERAGE should take average of the odd numbers in the range (and between the upper and lower limits). Similarly, if it is 2 then the CUSTOMAVERAGE should take the average of the even numbers in the range (and between the upper and lower limits)

Exercise 2: String Reverser

Below we will look at a program in Excel VBA that can reverse strings.

	A	B	C	D	E	F	G	H	I
1									
2									
3		Reverse String							
4									
5									

Place a command button on your worksheet and add the following code lines:

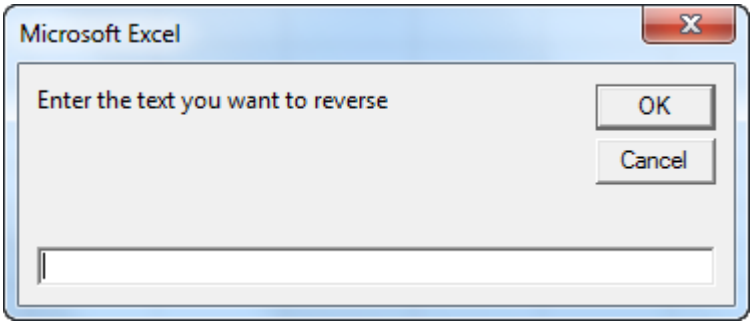
1. First, we declare four variables. One variable called text of type String, one variable called reversedText also of type String, one variable called length of type Integer, and one variable called i of type Integer.

```
Dim text As String, reversedText As String, length As Integer, i As Integer
```

2. We initialize two variables. We use the InputBox function to get a text string from the user. We use the Len function in Excel VBA to get the length of a string.

```
text = InputBox("Enter the text you want to reverse")

length = Len(text)
```



3. We start a For Next loop.

```
For i = 0 To length - 1
```

4. Now comes the simple trick. We take the last character from text and place it at the front of ReversedText. We can use the Mid function in Excel VBA to extract a character from a string. We use the & operator to concatenate (join) two strings.

```
reversedText = reversedText & Mid(text, (length - i), 1)
```

5. Don't forget to close the loop.

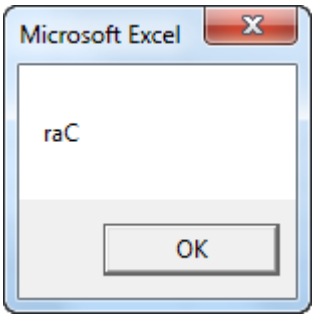
```
Next i
```

Example: text = "Car". The length of text is 3. For i = 0 to 2, we extract the substring of text starting at position length - i with length 1. Thus, for i = 0, Mid(text, 3, 1) equals r. We place r at the first position of reversedText. For i = 1, Mid(text, 2, 1) equals a. We add a to reversedText which becomes ra. For i = 2, Mid(text, 1, 1) equals C. We add C to reversedText which becomes raC.

6. Finally, we display reversedText using a MsgBox.

```
msgbox reversedText
```

7. Test the program.



Now, utilize this code to determine if the input string is palindrome or not. A string is palindrome if it reads the same from beginning to end and end to beginning. For example level, radar, civic are all palindrome since reversing them will not change the word. Your program should input the word and inform the user whether the word is palindrome or not. This should be case sensitive.

Exercise 3: Bills and Coins

Below we will look at a program in Excel VBA that splits an amount of money into bills and coins.

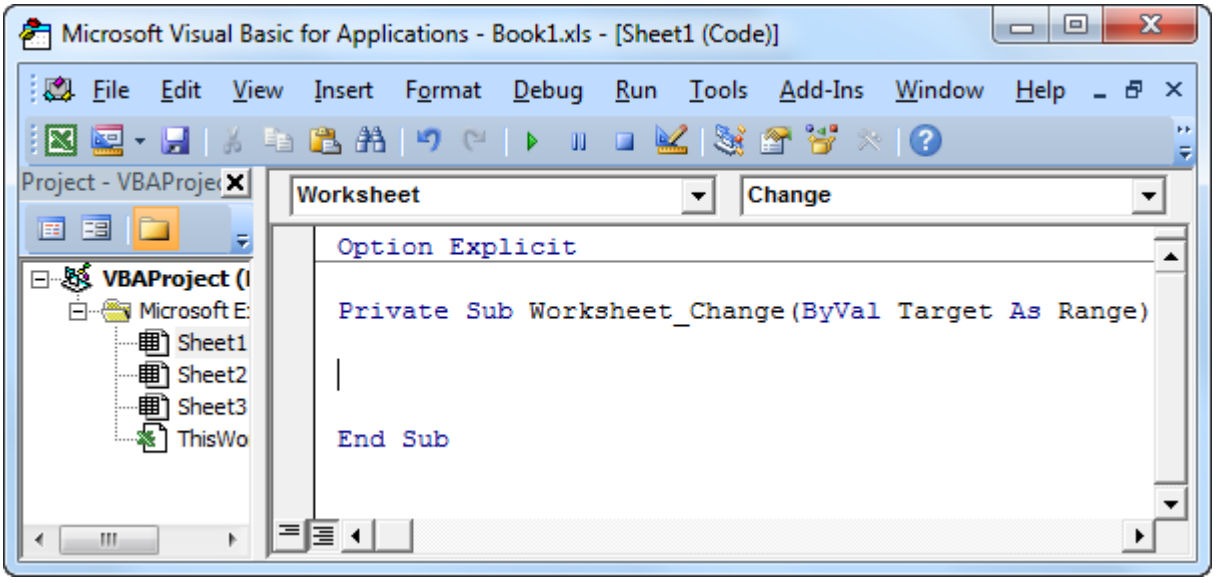
	A	B	C
1			
2	Amount		
3			
4	Bills/coins	Frequency	
5	$100.00		
6	$50.00		
7	$20.00		
8	$10.00		
9	$5.00		
10	$2.00		
11	$1.00		
12	$0.50		
13	$0.25		
14	$0.10		
15	$0.05		
16	$0.01		
17			

Create a Worksheet Change Event. Code added to the Worksheet Change Event will be executed by Excel VBA when you change a cell on a worksheet.

1. Open the Visual Basic Editor.

2. Double click on Sheet1 (Sheet1) in the Project Explorer.

3. Choose Worksheet from the left drop-down list. Choose Change from the right drop-down list.



Add the following code lines to the Worksheet Change Event:

4. Declare a variable called amount of type Double and a variable i of type Integer.

```
Dim amount As Double, i As Integer
```

5. The Worksheet Change Event listens to all changes on Sheet1. We only want Excel VBA to do something if something changes in cell B2. To achieve this, add the following code line:

```
If Target.Address = "$B$2" Then
```

6. We initialize the variable amount with the value of cell B2.

```
amount = Range("B2").Value
```

7. We empty the range with the frequencies.

```
Range("B5:B16").Value = ""
```

8. Now it's time to split the entered amount of money. We start a For Next loop.

```
For i = 5 To 16
```

9. We will make use of the Do While Loop structure. Code placed between these words will be repeated as long as the part after Do While is true. We want Excel VBA to repeat the code lines at step 10 as long as amount is larger or equal to `Cells(i,1).value`.

```
Do While amount >= Cells(i, 1).Value

Loop
```

10. Add the following code lines to the Do While Loop.

```
Cells(i, 2).Value = Cells(i, 2).Value + 1
amount = amount - Cells(i, 1).Value
```

Explanation: as long as amount is larger or equal to `Cells(i,1).value`, the amount contains bills/coins of this value. As a result, Excel VBA increments the frequency of this bill/coin (first line) and subtracts the value of the bill/coin from amount (second line). This process will be repeated until amount becomes smaller than `Cells(i,1).value`. Next, Excel VBA increments i and goes to the next bill/coin to see how many times this bill/coin fits in the amount left. This way the amount of money will be split into bills and coins until there is no money left to split anymore.

11. Close the For Next loop and don't forget to close the if statement (both outside the Do While Loop).

```
   Next i
End if
```

12. Test the program.

	A	B	C
1			
2	Amount	6675.74	
3			
4	Bills/coins	Frequency	
5	$100.00	66	
6	$50.00	1	
7	$20.00	1	
8	$10.00		
9	$5.00	1	
10	$2.00		
11	$1.00		
12	$0.50	1	
13	$0.25		
14	$0.10	2	
15	$0.05		
16	$0.01	4	
17			

Note: Of course, the entered amount does not necessarily contain every bill/coin. If amount does not contain a certain bill/coin, the part after Do While never becomes true for this bill/coin and Excel VBA goes directly to the next bill/coin.

Now, modify this code so that you have an upper limit on the number of each bill/coin to use. Set column C to keep the maximum amount of a denomination to use. For instance for the previous example if there is a limit of 60 $100 bills, then the new distribution is supposed to be like in the figure on the right. Your code should output a message box stating not possible, if it cannot find enough money to cover the amount given in B2.

Note that in your solution you might observe a rounding error of about 1 cent.

	A	B	C	D
1				
2	Amount	6675.74		
3				
4	Bills/coins	Frequency	Max	
5	$100.00	60	60	
6	$50.00	13	14	
7	$20.00	1	10	
8	$10.00		20	
9	$5.00	1	10	
10	$2.00		0	
11	$1.00		0	
12	$0.50	1	3	
13	$0.25		3	
14	$0.10	2	2	
15	$0.05		3	
16	$0.01	4	4	
17				
18				
19				