Lookup and If structures in UDFs

- Last week we saw how If structures can also be used inVBA codes for user-defined functions.
- We have also seen that every Excel built-in function can in principle be used inside a VBA programme. For this we need to refer to it as:

Application.WorksheetFunction.NAME(arguments)

- Today we will combine all of this by seeing examples of VBA user-defined functions that use the VLOOKUP and HLOOKUP Excel built-in functions in combination with VBA IF structures.
- VLOOKUP and HLOOKUP functions are used in user-defined functions' codes when the input values for the UDF are to be read from some table on the Excel spread sheet.
- There are many practical situations in which this may be useful. We will see this through several examples today!

- Recall the first exercise on the last Lab sheet (Lab 6).
- You should have written down a code that computes the “body mass index” given the weight and height of a person in the following way:

```
Function bmi(we as Single, he as Single) as Single
bmi= we/ (he) ^ 2
End Function
```

In the Excel WS we would get: =bmi(70, 1.71) → 24

- We could now use this function together with a table of weights and heights to produce a new function that takes the name of a person as input and gives their bmi as output.
- Suppose we have the following table on the Excel WS:

	A	B	C
1	Names	Weight (Kg)	Height (m)
2	A. Myles	67,2	1,7
3	S. Bennett	56,3	1,6
4	F. Taylor	45,09	1,52
5	R. Ruiz	89,2	1,75
6	A. Sas	67,7	1,55

If the name inputed does not exactly match one of the names in the table, the function returns an error message!

The function's input is a person's name

Range where the table is located

```
Function people(na As String) As Single
Dim w, h As Single
w = Application.WorksheetFunction.VLookup(na, [a2:c6], 2, False)
h = Application.WorksheetFunction.VLookup(na, [a2:c6], 3, False)
people = w / h ^ 2
End Function
```

The Vlookups extract the weight (column 2) and height (column 3) for a given name from the table

•Another sort of UDF that we could write using Lookup functions would be a function that takes someone's bmi as input and returns a message as output, depending on the value (you wrote a code for such function also in the last Lab).

•We will do this by taking the table below as starting point:

male	female	
<20	<19	underweight
20-24.9	19-23.9	normal weight
25-29.9	24-28.9	overweight
30-39.9	29-38.9	obese
≥40	≥39	extreme obese

=Vlookup(27,B10:C14,2)

→ overweight

=Vlookup(36,B10:C14,2)

→ obese

	A	B	C
1	male	female	
2	0	0	underweight
3	20	19	normal weight
4	25	24	overweight
5	30	29	obese
6	40	39	extreme obese

The table that we need to enter in the Excel WS would rather look like this

- The two functions below would solve the problem:

```
Function female(bmi As Single) As String
female = Application.WorksheetFunction.VLookup(bmi, [b11:c15], 2)
End Function

Function male(bmi As Single) As String
male = Application.WorksheetFunction.VLookup(bmi, [a11:c15], 3)
End Function
```

- The first one works when the input bmi is that of a woman and the second one corresponds to the case of a man.
- Notice that the only difference is the table that the Vlookup function is looking at!
- As you did last week in the lab, it is not difficult now to write down the code for a function that combines these two together and takes as input both the bmi and the gender.

```
Function both(bmi As Single, ge As String) As String
If ge = "male" Then
both = Application.WorksheetFunction.VLookup(bmi, [a11:c15], 3)
ElseIf ge = "female" Then
both = Application.WorksheetFunction.VLookup(bmi, [b11:c15], 2)
Else
both = "I don't know about this gender"
End If
End Function
```

- Alternatively, if the functions *male* and *female* are in the same module as the function *both*, you could also write the latter as:

```
Function both(bmi As Single, ge As String) As String
If ge = "male" Then
both = male(bmi)
ElseIf ge = "female" Then
both = female(bmi)
Else
both = "I don't know about this gender"
End If
End Function
```

- Let us look at example from the January 2008 exam:

4) Consider the table given below and suppose that you enter the data on this table into the range A1:F4 of your Excel worksheet:

	Parmar	Lopez	Hirch	Oota	Patel
W1	41	43	42	43	45
W2	45	46	40	50	45
W3	50	41	48	42	35

The first row of the table contains the names of several employees of a company. The second, third and fourth rows contain information about the number of hours that each of this employees has worked in the first, second and third weeks after the start of their contract, respectively. This exercise has three parts:

i) (10 points) Write the code of a user defined function called **avehours** that takes the name of a given employee as input produces the average number of hours per week that he/she has worked over the first three weeks of his/her contract as output. The program should employ the worksheet function **HLOOKUP** and all variable types (both for the input and the output of the function should be defined)

	A	B	C	D	E	F
1		Parmar	Lopez	Hirch	Oota	Patel
2	W1	41	43	42	43	45
3	W2	45	46	40	50	45
4	W3	50	41	48	42	35

- We need a function that computes the average number of hours worked for a given name.
- For example, if the name is “Parmar”, then the function should return (41+45+50)/3 and similarly for the rest.
- A code that achieves this is:

```
Function avehours(x As String) As Single
Dim y1 As Integer
Dim y2 As Integer
Dim y3 As Integer
y1 = Application.WorksheetFunction.HLookup(x, [a1:f4], 2, False)
y2 = Application.WorksheetFunction.HLookup(x, [a1:f4], 3, False)
y3 = Application.WorksheetFunction.HLookup(x, [a1:f4], 4, False)
avehours = (y1 + y2 + y3) / 3
End Function
```

- The remaining two questions for this exercise were:

ii) (10 points) The second function should be called **class** and take again as an input the name of a certain employee. The function **class** should make use of the function **avehours** defined in i) and employ an **IF...ELSEIF** structure. Depending on the name of the employee, the function should return one of the following values:

- If the average number of hours of work per week is lower than 42 it should return the message: "not enough work"
- If the average number of hours of work per week is bigger or equal 42 and smaller than 47 it should return the message: "decent performance"
- If the average number of hours of work per week is larger than 46 then it should return the message: "to be recommended for promotion"

iii) (5 points) Compute avehours("Lopez") and class("Lopez").

- The important things to notice for part ii) is that we need to define a new function which still takes the name of an employee as input. The output is one of the three given messages (so, it is of string type as well).
- The function should use the function defined before to compute the average amount of hours worked. A code for this would be…

```
Function class(x As String) As String
Dim y As Single
y = avehours(x)
If y < 42 Then
class = "not enough work"
ElseIf y >= 42 And y < 47 Then
class = "decent performance"
Else
class = " to be recommended for promotion"
End If
End Function
```

- Important things: definition of variable types, declaration of the variable y (average number of hours worked), IF…ELSEIF structure.
- For question iii) we just had to evaluate both functions for the name “Lopez” and this gives:

=avehours(“Lopez”) → 43,333

=class(“Lopez”) → “decent performance”