MS office offers a functionality of building complex actions and quasi-programs by means of a special scripting language called VBA (Visual Basic for Applications). In this lab, you will learn how to use the Macros module and get familiar with basic syntax and formulation of VBA in the case of Excel. Note that getting acquainted with VBA programming is central for your pair-project.

BASIC MACROINSTRUCTIONS (MACROS)

Exercise 1

MS Office offers a wide variety of built-in macros that can facilitate edition or customization of tacks in both Excel and Word. All are programmed in VBA but their code is not accessible. However, it is a good practice to test one of these to get a view on the possibilities offered for future programming. The Macros that is proposed to be tested in part of Word: the Mail Merge Manager (MMM). Note that you may find some interest in using such macro for you project. Just to make it clear: it is expected NOT TO BE USED FOR YOU OWN PAIR-PROJECT!

How to proceed. Very simple:

1. Open a blank document in Word and save it as VBA1_*yourname*.docx
2. Open a blank document in Excel and save it as VBA1_*yourname*.xlsx
3. In Excel and first spreadsheet (sheet1. You can rename it "ex1" if you wish), enter some name of people that you would like to invite, for instance, for Your birthday (e.g. column A: firstname, column B: last name). Max. 10 entries. Save and go back to word. To remember the category, write in cell A1, "firstname" and in cell A2, "lastname". You will see that this will be also very useful later on.
4. Go back to Word and write some lines of text for your birthday invitation. The purpose of MMM is to generate X instances of a document in which some fields (e.g. names of invited persons) will be automatically changed based on the Excel entries.
5. Click now on the **Mailings** tab, then select **Step by Step Mailing Merge Wizard...**
   - **a)** Choose the type of document you want to create. In our case, select **Letters**.
   - b) Click **Next: Starting document** to move to Step 2. Then, Select **Use the current document**
   - c) Click **Next: Select recipients** to move to Step 3. This step simply links with the document where are located the list of interest (in our case names in excel document). So in the new window, click on **Browse...** since your list already exists in VBA1_*yourname*.xlsx.
   - d) Select the spreadsheet where are located the data. Also, tick option **First row of data contains column header** (see previous point 3). Press OK, then in new window uncheck any list member that you don't want to use in the final merging document. Press OK.
   - e) Step 4. Place fields where they will appear in your text, e.g. firstname and lastname. For that, place the cursor in the appropriate place of you text, then click in the MMM window on **More items...** and select the fields.
   - f) After it is just a matter of preview the results and finish the merging to be ready for printing or mailing.

As already mentioned, we cannot access to the code that is behind the MMM. Fortunately, this is not always the case as it is possible in MS Office to record some macros and later on see their corresponding VBA codes. The 2 next exercises are intended to illustrate such approach.

Exercise 2

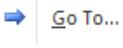
1. Enter the following data to a worksheet (e.g. in sheet2 of VBA1_*yourname*.xlsx):

location	books	papers	stamps	postcards	total
Łódź	11	35	10	10	**66**
Radomsko	13	24	12	12	**61**
Sieradz	21	8	28	28	**85**
Zgierz	8	2	16	26	**52**
total	**53**	**69**	**66**	**76**	**264**

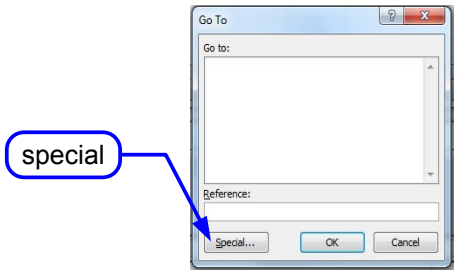
formula

Ensure, that the total values are calculated using the SUM function.

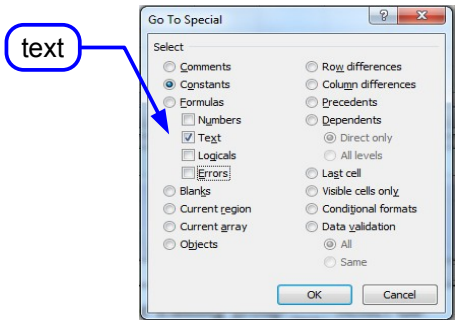
2. Formatting procedure:
   - select any single cell
   - click the **Home** tab, then click on the **Find&Select** button in the Editing group 

. Select **Go To...** 

   - in a **Go To** dialog choose **Special** button:



   - Select **constants** option and leave unchecked options **Numbers**, **Logicals** and **Errors** (only **Text** is checked):



   - click **OK**
   - format the selection as follows:

location	books	papers	stamps	postcards	total
Łódź	11	35	10	10	**66**
Radomsko	13	24	12	12	**61**
Sieradz	21	8	28	28	**85**
Zgierz	8	2	16	26	**52**
total	**53**	**69**	**66**	**76**	**264**

   - rrepeat the steps for **Formulas** resulting in **Numbers** and apply the format:

location	books	papers	stamps	postcards	total
Lodz	11	35	10	10	66
Radomsko	13	24	12	12	61
Sieradz	21	8	28	28	85
Zgierz	8	2	16	26	52
total	53	69	66	76	264

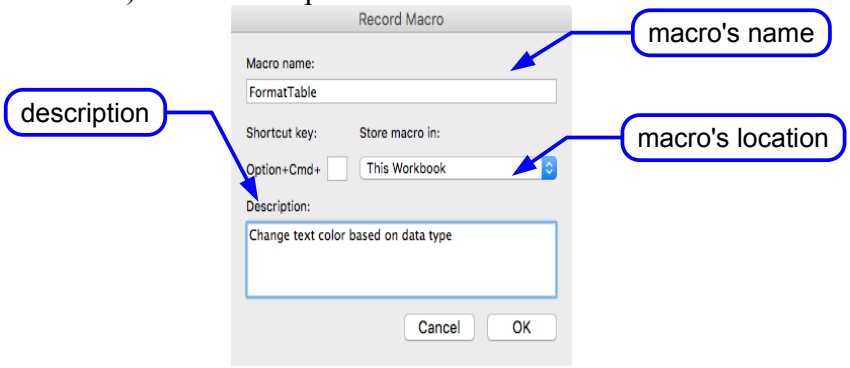
   - repeat the steps for **Constants - Numbers** and apply the format. Then, add the legend (**add new rows** and place the legend cells there):

	A	B	C	D	E	F
1		text				
2		formulas				
3		numbers				
4						
5	location	books	papers	stamps	postcards	total
6	Lodz	11	35	10	10	66
7	Radomsko	13	24	12	12	61
8	Sieradz	21	8	28	28	85
9	Zgierz	8	2	16	26	52
10	total	53	69	66	76	264

Exercise 3

Knowing, how to format the cells, you can start recording a macro.

1. Select the whole worksheet (e.g. pressing **Ctrl+A**) and then clear the formatting (click on **Clear** in the **Editing** group on **Home** tab. Select **Clear Formats** from the list). **Delete the three first rows** containing the legend.
2. Macro recording procedure:
   - select **Macros** on the **View** Tab → **Record Macro...**
   - call the macro with any name (but I propose to give a meaningful name, e.g. FormatTable), add the description and select "This Workbook" or the name of your file (e.g. VBA1_*yourname*.xlsx) from the dropdown list which macro's location:



   - click **OK**
   - repeat all the steps performed in the previous exercise
   - when finished, choose **Stop Recording** from the list displayed after clicking on the **Macros** button
   - in order to test the macro, again remove the worksheet format. Choose **View Macros**. Select your macro and press **Run**.
   - **Save As** your file and change the extension to .xlsm (e.g. VBA1_*yourname*.xls**m**). Why? Because your file now contains VBA macros (**m** stands for macro).

Exercise 4

Recorded macro very often needs some modifications. In order to edit the macro, you must find the worksheet containing it. In our case, we have chosen "this workbook" when creating macros, so it is enough to open macro dialog, select the macro and press **Edit** button.
Visual Basic Editor window opens, while the standard MS Excel windows still remains open. If you want to close the editor, use **File** menu → **Close and return to Microsoft Excel** (or just close the editor).

Before analyzing the code corresponding to the recorded macros and doing quite advanced VBA operations, let get acquainted with the basic syntax. First of all, you will find in the **Annex** at the end of this document tables containing most of the basic VBA syntax and statement structures.

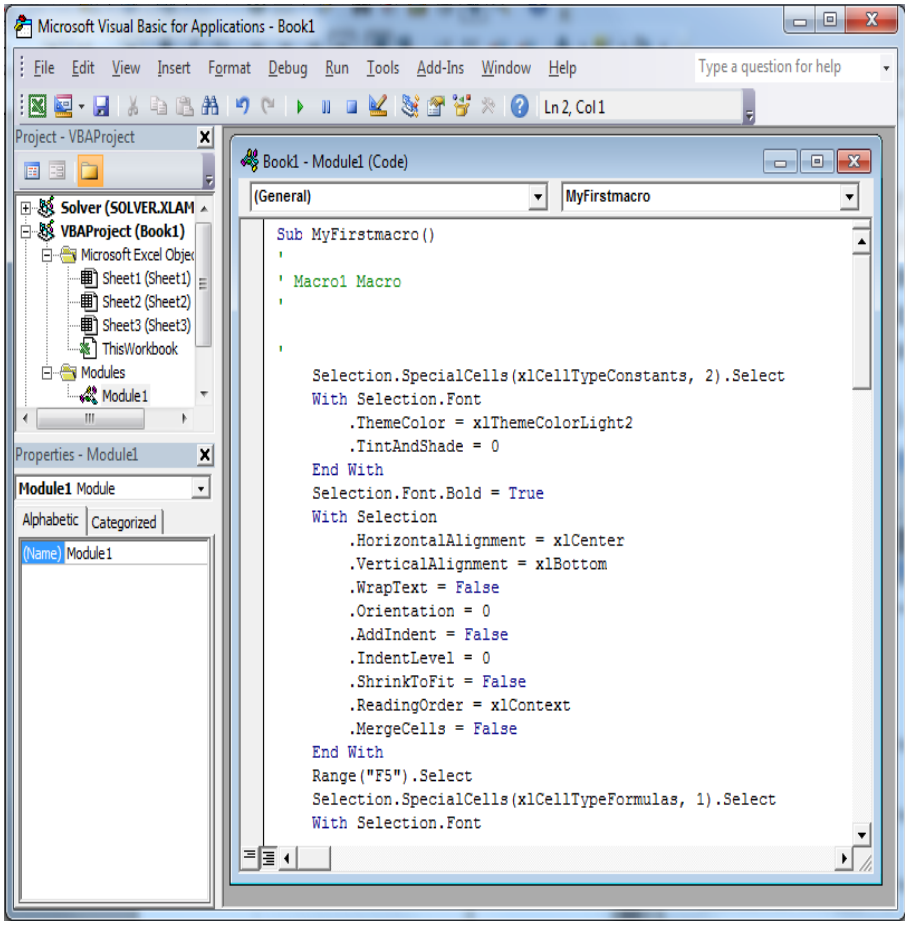
Most of them will be detailed during the VBA labs. Then, let run few basic macros and create new ones to practice coding. For that, open the **Visual Basic editor** if not done yet. In the *VBA project* window (left), look at the *Modules* folder under your excel file. There should be **Module1**, which contains the recorded code corresponding to the formatting exercise. Click right on Modules and select Insert → Module. **Module2** should appear. Now, open the text provided as a link in the webpage and copy the content inside **Module2**.
Execute provided codes (put cursor in the code and press play button ) and follow instructions.Note the spreadsheet should look like the last illustration of Exercise 2.

Exercise 5

Go now to **Module1** where recorded procedures of Exercise 3 are.
The internal frame contains the code of your macro (note that it may look different than the example shown below). Notice, that the lines prefixed with apostrophe " ' " are not interpreted – they are the **comment lines** displayed in green (see also Annex). The comments (code documentation) are very important for the code conservation and maintenance. They also allow to "switch off" the code fragments that we do not want to execute, but they may be useful later. The comment explaining the code line meaning should look as follows:

```
ColorIndex = 5 'changes colour to  blue
```

while the explanation for the whole code block should be placed as a separate line before the block.



Add a few comments:

- find the line:
  `Selection.SpecialCells(xlCellTypeConstants, 2).Select`
  add a new line before. Place there your comment (remember about the apostrophe):

```
'find and format the cells containing text
```

- do the same with:
  `Selection.SpecialCells(xlCellTypeConstants, 1).Select`
  adding the comment about the number cells
- again, for:
  `Selection.SpecialCells(xlCellTypeFormulas, 1).Select`
  add the comment about cells containing formulae resulting in numbers
- find the instruction (the row number may be different):
  `Rows("1:1").Select`
  and add a comment above about code block responsible for creating the legend.

The code above shows you some characteristics of VBA programming. One main attribute is the use of so-called VBA objects that are specific for MS environments, i.e. Word, Excel of even PowerPoint. For instance, you can see above that **Range**, **Selection** or **Rows** are typical Excel objects. VBA objects are characterized by ***attributes*** (or properties, e.g. **Font**, **Orientation**) and ***methods*** (or functions, e.g. **Select**). When you change their characteristic in VBA, their appearance and content may change in the MS environment. You will learn more about objects and hierarchy of objects in the next sections.

Exercise 6

Record macro to clear format and delete the 3 rows of legend. Name it **ClearFormat**.

ANNEX – VBA basic syntax

General

Line continuation character	_ (underscore)
Comments	‘ (single quote) or REM …

Procedure

Sub routine	SUB SubName (ParameterList) ... END SUB
Function	FUNCTION FunctionName (ParameterList) AS type … END FUNCTION
Scope	PUBLIC PRIVATE

Variables

Variable names	Not a keyword <= 255 characters First character must be a letter Cannot contain a period, space, !, @, #, &, %, or $
Variable declaration	DIM VariableName AS type
Object variables	DIM ObjectVariable AS ClassName e.g., DIM frmAny AS form
Create new objects	SET ObjectVariable = NEW ClassName
Frees up memory associated with objects	SET ObjectVariable = Nothing
Arrays	DIM ArrayName(size) AS type DIM ArrayName(1 TO #) AS type

Constants

Symbolic constant	CONST constant_name = value [AS type] e.g., CONST path = “d:\”

Operators

Assignment operator	VariableName = expression SET Obj1 = Obj2
Mathematical	+ - * / \ (integer division with rounding) ^ mod (remainder after division)
String	& (concatenation)
Comparison	= > < >= <= <> (different than)
Logical	And Or Not

Conditional control

If … Then … Else	IF condition THEN
	  statements executed under true condition
	[ELSE
	  statements executed under false condition
	END IF]

Select Case	SELECT CASE test-expression
	  CASE expression-list-1
	  statement-block-1
	  …
	  CASE expression-list-n
	  statement-block-n
	END SELECT

Loop

For … Next	FOR counter = start TO end [STEP increment] statements NEXT
For Each … Next (a collection of objects)	FOR EACH item in collection statements NEXT item
With … End With (a single object)	WITH object statements accessing the properties/methods of the object END WITH
Do While … Loop	DO WHILE condition statements executed as long as the specified condition is true LOOP
Do Until … Loop	DO UNTIL condition statements executed until the specified condition is true LOOP

Unconditional control

GoTo	GOTO label … … label:
Exit a procedure	EXIT
Exit a For loop	EXIT FOR
Exit a Do loop	EXIT DO