EXERCISE 3a (transposition cipher in letters mode)

Symmetrical cryptography

The first known methods of encryption were not too complicated, but were sufficient for the current needs. Symmetrical encryption was the most popular group of ciphers, where encryption and decryption was based on one and the same key. The only required need was knowing the algorithm and the key. The system security was depended on the secrecy of the key, so it was necessary to find out the way of secure exchange it between the sender and receiver.
Symmetrical systems may be divided on two fundamental types: **transposition** and **substitution**.

A **transposition cipher** is a method of encryption by which the positions held by units of plaintext are shifted according to a regular system, so that the ciphertext constitutes a permutation of the plaintext. That is, the order of the units is changed. The efficiency of this method depends on both, the algorithm itself and the length of the plaintext – the longer the text, the bigger number of permutation of signs.

A **substitution cipher** is the data encryption scheme in which units of the plaintext (generally single letters or pairs of letters of the plaintext text) are replaced with other symbols or groups of symbols.

There is also another division of symmetrical cryptography: **mono-** and **polyalphabetic**. According to the name, a polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets. The Vigenère cipher is probably the best-known example of a polyalphabetic cipher, though it is a simplified special case.

Ciphers

A transposition cipher

The principle of this kind of encryption is shown here:

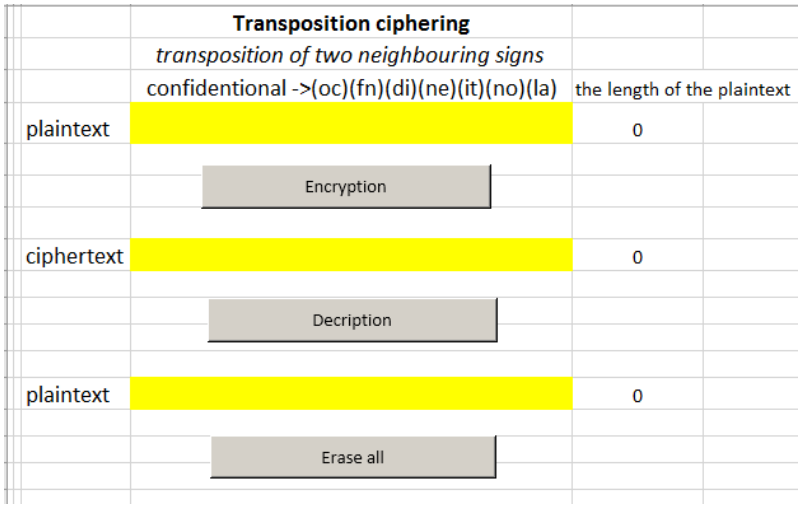
plaintext/public text/	*confidentional*
secret text/private text/ciphered text/encrypted text	*ocfndineitnola*

May one can see that the ciphered text was created from the original one by transposition of two neighbouring letters. The way of transposition depends on arbitrary choice of the author but the receiver must be informed how it works.

Performing the exercise

We will be using the Excel spreadsheet and Visual Basic programming for exercising this type of cipher in both: encryption and decryption purposes. There will be necessary to use of **macro** idea, which is an action or a set of actions that you can run as many times as you want. When you create a macro, you are recording your mouse clicks and keystrokes.

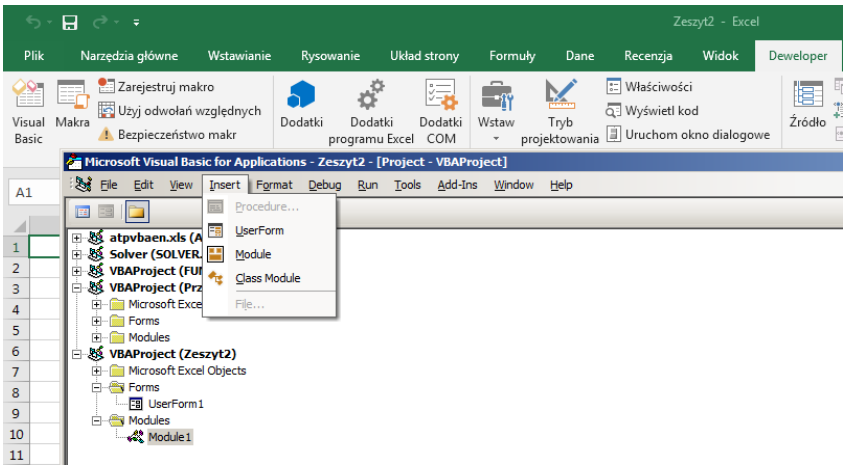
At the beginning you must prepare the spreadsheet similar to the shown below:



Continue following steps:

– Insert the Excel function *length()* into cells containing 0 value. It will calculate the number of signes (letters) composing the plaintext and ciphertext in yellow bars to the left. Notice, that all yellow bars represent one Excel cell of longer size than the average one.
– In order to support the spreadheet with the buttons is necessary to add the **Developer** to the Excel toolbar.

All functions and subroutines must be placed within one common VBA **Module**, used to store every product of the VBA Editor.



User has to follow this way: **Developer Tab** from the **Toolbar** ribbon → **Visual Basic** option → **Insert** → **Module** from pull down menu that are listed in order as **Module1**, **Module2** etc.

If the **Developer** tab in your Office application is not visible, then

– Click the **File** tab.
– Click **Options**.
– Click **Customize Ribbon**.
– Under **Customize the Ribbon** and under **Main Tabs**, select the **Developer** check box.

Encryption

The principle of encryption algorithm is shown as the flowchart in Fig. 1

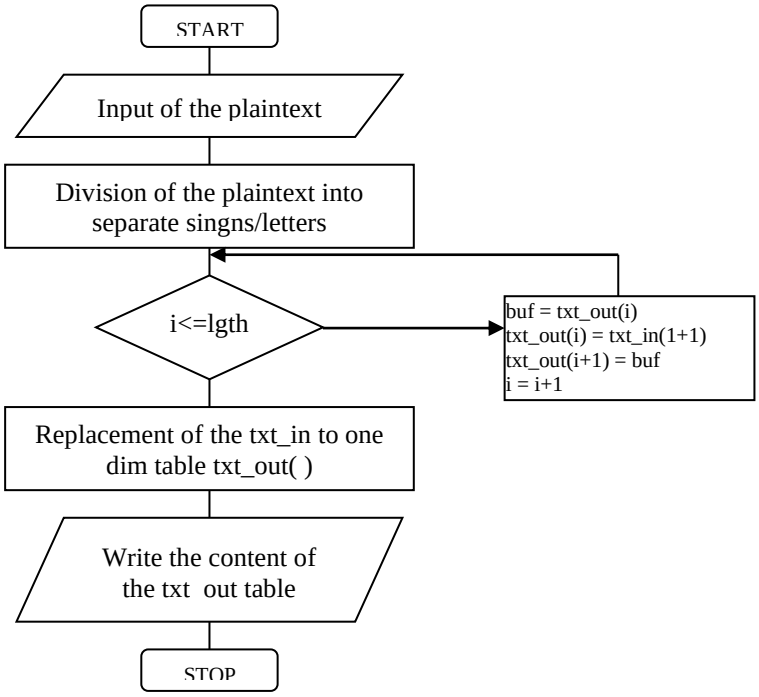


Fig. 1. A flowchart of transposition algorithm

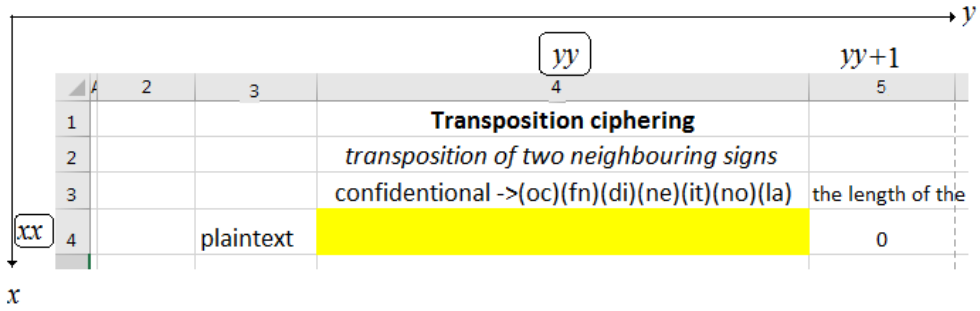
It is composed of two separated parts. The first one works as a VBA function *div_txt* that divides input text into single signs/letters.

```
Dim txt_out(1000) As String
Function div_txt(txt_in, lgth As Integer) As String
For i=1 to lgth
  txt_out(i) = Mid(txt_in, i, 1)
Next i
divtxt=txt_out(i)
End Function
```

where: `lgth` – the length of the `txt_in`,`txt_in` – string containing input plaitext before signes separation; `txt_out` – the output text composed of single signs/letters and stored in one dimentional table (one sign/letter – one cell); `MID(text, start_num, num_chars)`– Excel internal function that extract **text** from inside a **string,** `MID` - returns a specific number of characters from a **`text string`**, starting at `start_num` and continuing through `start_num + num_chars`.

As `div_txt` is a VBA function, it must be called in the following way:

```
a = div_txt(Cells(xx, yy), Cells(xx, yy+1))
```



where: `xx`, `yy` is the address of cell containing plaintext (coordinates differs for every students and are given individually), `xx`, `yy+1` – the address of cell with the length of plaintext

The second part of algorithm works as VBA subroutine that replaces signs/letters: `co -> oc`, `nf -> fn` within pairs of them. This code must be added to the previous one.

```
Sub encryption()
Cells(xx+9,yy) = "" ' empty cell
Erase txt_out ' embedded VB function - erasing content of the table
a = div_txt(Cells(xx+4, yy), Cells(xx+4, yy+1)) 'auxiliary variable
For i = 1 To Cells(xx+4, yy+1) Step 2 'by every second step
 buf = txt_out(i)
 txt_out(i) = txt_out(i+1)
 txt_out(i+1) = buf
Next i
For i = 1 to Cells(xx, yy+1)+1
 Cells(xx+4,yy) = Cells(xx+4, yy)&txt_out(i)
Next i
End Sub
```

Decryption

Procedure of decryption is similar to encryption but it works in reverse direction. The only difference is in addressing cells where are the input and output tables.

```
Sub decryption()
Cells(xx+9, yy) = ""
Erase txt_out
a = div_txt(Cells(xx+4, yy), Cells(xx+4, yy+1)) 'auxilary variable
For i = 1 To Cells(xx+5, yy+1) Step 2 'transposition by every 2 signs
 buf = txt_out(i)
 txt_out(i) = txt_out(i + 1)
 txt_out(i + 1) = buf
 Next i
For i = 1 To Cells(xx+4, yy+1)+1
 Cells(xx+9, yy)= Cells(xx+9, yy)&txt_out(i)
Next i
End Sub
```

To complete the algorithm all cells will be cleaned by means of the following subroutine

```
Sub Erase_Cells()
Cells(xx, yy) = ""
Cells(xx+4, yy) = ""
Cells(xx+9, yy) = ""
End Sub
```

Do not forget to add one important line of general declarations at the top of the **Module** body:

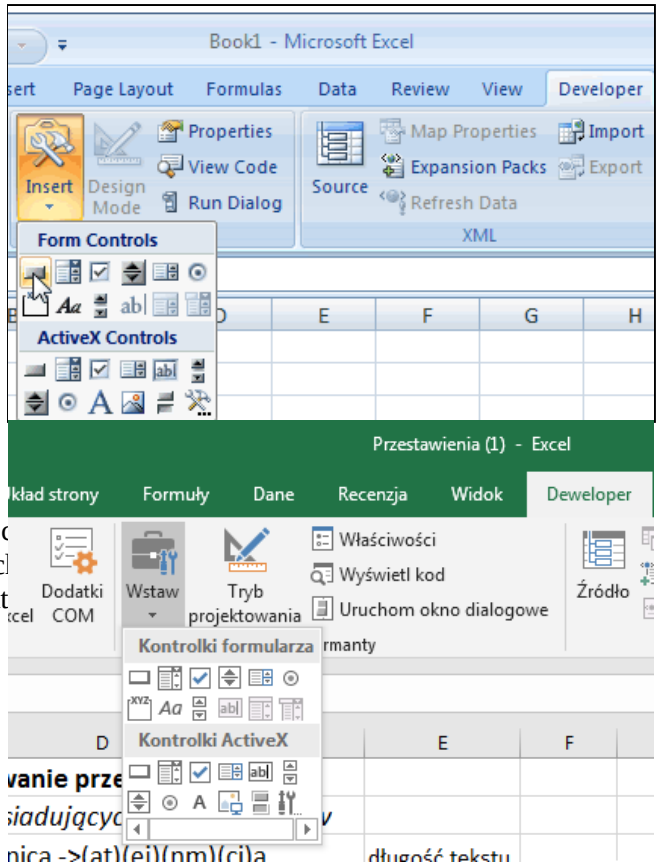
```
Dim txt_out(1000) As String
```

what is necessary if the access to this table has to be available within the functions and subroutines belonging to the common **Module**.

Now is time to add three buttons to the Excel sheet. It it will be done by means of Open VBA editor for creation the **Macro** modules associated with all buttons: *Encryption*, *Decryption* and *Erase all*.

Assigning a **Macro** to a **Form** or a **Control button**

To add a button (**Form controls**) on the **Developer** tab, in the **Controls** group, click **Insert**, and then under **Form Controls**, click **Button**. Click the worksheet location where you want the upper-left corner of thebutton to appear. Assign a macro to the button, and then click OK. To change the name of the Button - right click twice mouse on it and type the name you want to have. When you repeat it again then assigning the button to the macro can be done. The dialog pops up in the active worksheet with a list of macros to be chosen by you.



Report should contain

1. Properly running program of transposition c
2. Pay attention to odd and even numbers of cl
3. How does the program work with the plaint

EXERCISE 3b (transposition cipher in binary version)

Computers in principle works on bits. Every single sign in ASCII code is represented by the sequence of 8 bits, what gives the great chance of making encryption by operations on bits more complicated and more difficult to be broken. It is quite possible to repeat similar way of encryption like in Exercise 3a but dealing only with bits.

A transposition cipher in binary version
The necessity of transfering ASCII code to bits significantly extends code of the program written in modules. An example of this idea is presented here:

plaintext: `code`
ciphertext: `01100011 01101111 01100100 01100101`

A transposition cipher in binary version

tranposition of two neighbouring bits in the plaintext

text->(10)(11)(10)(00) (10)(01)(10)(10) (10)(11)(01)(00) (10)(11)(10)(00)

Plaintext	NEVERFOREVER	12		
	Encryption		workspace for ASCII<-> BIN conversion	
			R	sign
Plaintext in bits	010011100100010101010110010001010101001000100011001001111010100100100010101010110010001010100010	96	82	decimal ASCII code
Ciphered text in bits	100011011000101010101001100010101010000110001001100011111010000110001010101010011000101010100001	12	01010010	binary ASCII code
	Decryption			
Recovered ciphertext in bits	010011100100010101010110010001010101001000110010011110101001001000101010101100100010101010010	96	01010010	binary ASCII code
	(10)(11)(10)(00) (10)(01)(10)(10) (10)(11)(01)(00) (10)(11)(10)(00)		82	decimal ASCII code
Recovered plaintext	NEVERFOREVER	12	R	sign
	Clear all			

And here is the code in VBA:

```
Dim txt_out(500) As String
Dim txt_ASCII_bin(10000), txt_ASCII_8chr(10000) As String
Function dziel_tekst(txt_in As String, dl As Integer) As String
For i = 1 To dl
  txt_out(i) = Mid(txt_in, i, 1)
Next i
  dziel_tekst = txt_out(dl)
End Function

Sub szyfracja_ASCII()
Erase txt_out, txt_ASCII_bin
a = dziel_tekst(Cells(4, 4), Cells(4, 5))
For i = 1 To Cells(4, 5)
 Cells(6, 6) = txt_out(i)
 txt_ASCII_bin(i) = Cells(8, 6)
Next i
Cells(7, 4) = ""
For j = 1 To Cells(4, 5)
Cells(7, 4) = Cells(7, 4) & txt_ASCII_bin(j)
Next j
Erase txt_out
b = dziel_tekst(Cells(7, 4), Cells(7, 5))
For i = 1 To Cells(7, 5) Step 2
 bufor = txt_out(i)
 txt_out(i) = txt_out(i + 1)
 txt_out(i + 1) = bufor
Next i
 Cells(8, 4) = ""
For j = 1 To Cells(7, 5)
 Cells(8, 4) = Cells(8, 4) & txt_out(j)
Next j
End Sub
```

```
Sub deszyfracja_ASCII()
Erase txt_out, txt_ASCII_bin
b = dziel_tekst(Cells(8, 4), Cells(8, 5) * 8)
For i = 1 To Cells(8, 5) * 8 Step 2
 bufor = txt_out(i)
 txt_out(i) = txt_out(i + 1)
 txt_out(i + 1) = bufor
 Next i
Cells(11, 4) = ""
For j = 1 To Cells(8, 5) * 8
 Cells(11, 4) = Cells(11, 4) & txt_out(j)
Next j
Erase txt_ASCII_8chr
k = 1
i = 1
Do While k <= Cells(8, 5) * 8
 txt_ASCII_8chr(i) = Mid(Cells(7, 4), k, 8)
 k = k + 8
 i = i + 1
Loop
Cells(13, 4) = ""
For i = 1 To Cells(11, 5) / 8
 Cells(11, 6) = txt_ASCII_8chr(i)
 txt_ASCII_bin(i) = Cells(13, 6)
 Cells(13, 4) = Cells(13, 4) & txt_ASCII_bin(i)
Next i
End Sub

Sub wytrzyj2()
Cells(4, 4) = ""
Cells(7, 4) = ""
Cells(8, 4) = ""
Cells(11, 4) = ""
Cells(13, 4) = ""
End Sub
```

Report should contain

1. Properly running program of transposition cipher with offset (xx, yy) assigned to each student and pointing the VBA address of the plaintext in yellow cell.
2. Pay attention to odd and even numbers of characters in the text.
3. How does the program work with the plaintext containing spaces?