Introduction to programming – ASSIGNMENTS 5

General instructions:

- Type in and test your programs using Python Idle.
- Bring in your answers to the next assignment (i.e. demonstration) session in memory stick, or save them to a web folder accessible in class. Alternatively, you can bring written answers with you, though this is not recommended.
- Remember to comment your code – this does NOT mean, that every single line should be commented. Comment the important parts in your program.
- Prepare to present your solution to the class.

1.

Write a procedure that gets two parameters, **n** and **m**, and proceeds to output a multiplication table with values $(1 \cdot n, 2 \cdot n, \dots, m \cdot n)$.

Example output (with argument n == 5 and m == 3):

1 * 5 = 5
2 * 5 = 10
3 * 5 = 15

2.

Write a procedure **`outputUnderlined(s)`** which outputs the given string argument **`s`** and proceeds to output a line made of hyphen characters ("-") to the next line. The line must be the same length than the output string.

Example run (in Python shell):

```
>>> outputUnderlined("Hello! ")
Hello!
------

>>> outputUnderlined("This is a test ")
This is a test
--------------
```

3.

Write a function **`isValid(string, minLen, maxLen)`** which returns True, if the length of the given string is between `minLen` and `maxLen`, i.e. $minLen \leq length \leq maxLen$, and False in all other cases.

Example run (in Python shell):

```
>>> print isValid("abcd", 2, 6)
True

>>> print isValid("Hello all", 4, 7)
False
```

4.

The **lowest common multiple (LCM)** of numbers *a* and *b* is the smallest positive integer that is a multiple of both *a* and *b*. For example, the LCM for numbers 4 and 6 is 12, and the LCM for number 5 and 10 is 10.

In other words, the LCM for numbers a and b is the smallest number, that is divisible with a and b.

Write a function LCM, that returns the lowest common multiple for two given arguments:

Example run (in Python shell):

```
>>> print LCM(3,5)
15

>>> print LCM(8, 10)
40
```

5.

Write two functions, rotateLeft(pattern) and rotateRight(pattern). The functions get a string containing a bit pattern of any length as an argument, and rotate that pattern one step left or right. The rotated pattern is then returned as a string.

In rotation all bits in pattern are moved one step left or right, and the bit that 'falls off' is moved to an empty space:

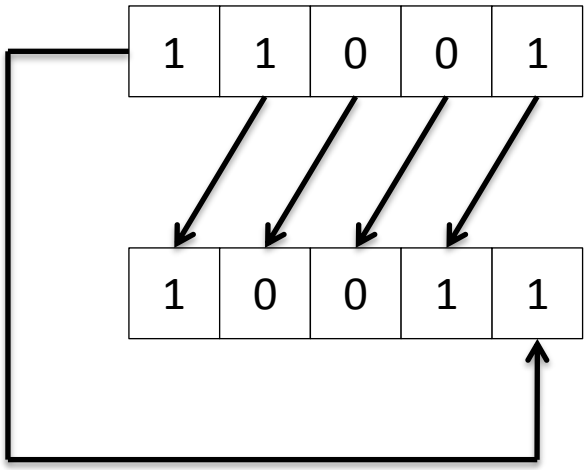

Figure 1: Rotating pattern 11001 one step left results to 10011

Using these two functions, write a program that queries the user for a bit pattern, a direction and number of steps, and proceeds to output the pattern rotated the given number of steps in that direction.

Example run:

Give a bit pattern: 100011
Give direction (left / right): right
Give number of steps: 3
Rotated pattern: 011100

6. *** Expert assignment (double points*

ROT-13 is a very simple cryptographic algorithm, which substitutes each letter (from A to Z) in a string with a letter 13 places further along in the alphabet (e.g. A is substituted with N, K with X and T with G).

Figure 2 demonstrates the use of the algorithm. Note, that ROT-13 is its own inverse, meaning that the same algorithm can be used to code and decode a message (or, `rot13(rot13(string)) == string`).

Write a function `rot13(string)`, which returns the string given as an argument in ROT-13 decoded form. You can assume, that the string contains lowercase letters from 'a' to 'z' only.

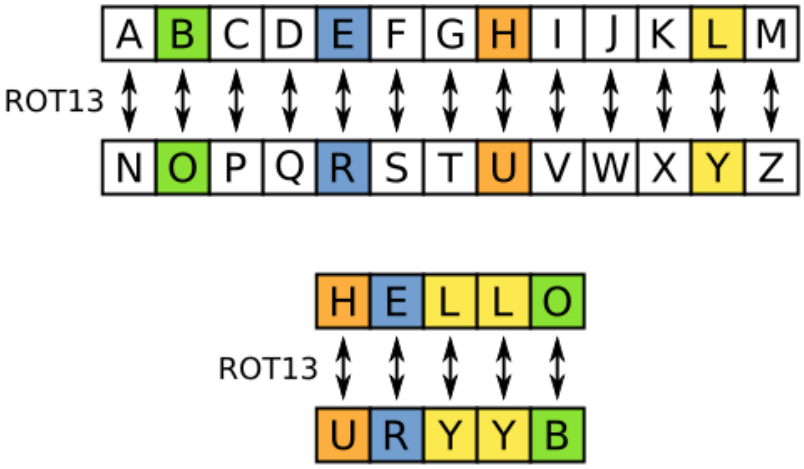

Figure 2: ROT-13 algorithm. Source: wikipedia.com