Exercises

Week 38

DM536 Introduction to Programming
DM562 Scientific Programming
DM857 Introduction to Programming
DS830 Introduction to Programming

1 Making Choices

1. Define a function `sign(n)` that returns the sign of the number n using the following algorithm;

$$sign(n) = \begin{cases} 1 & \text{if } n > 0 \\ 0 & \text{if } n = 0 \\ -1 & \text{if } n < 0 \end{cases}$$

2. Define a function `to_meters(length,unit)` that converts `length` given in `unit` to meters using the following table (the unit is expressed using strings in its extended or abbreviated form).

	unit		meters
1	`'inch'`, `'in'`	=	0.0254
1	`'hand'`, `'h'`	=	0.1016
1	`'foot'`, `'ft'`	=	0.3048
1	`'yard'`, `'yd'`	=	0.9144

For instance `to_meters(30,'in')` must return `0.762`.

3. Write a function `print_conversion_table(length)` that prints a table (like the one above) with the conversion to meters of `length` if this value is taken in inches, hands, foots, or yards. For instance, `print_conversion_table(30)` prints the following text (you may change the format).

```
30in = 0.762m
30h = 3.048m
30ft = 9.144m
30yd = 27.342m
```

4. Write a program to compute the perimeter and area of a square. The program starts by asking the user to input the side of the square (assume it is a floating point number). If the input is positive, then it prints the perimeter and the area. Examples:

```
Enter the side of the square (a positive number): 5.0
The perimeter of a square of side 5.0 is 20.0.
The area of a square of side 5.0 is 25.0.
```

If the input is not positive, then it prints a message saying that the input should be positive and terminates.

```
Enter the side of the square (a positive number): -1.0
The value for the side must be a positive number; -1.0 is not positive.
```

If the input is not a number, then the program terminates with an error.

5. Write a program to compute the area of circles, rectangles, squares, and triangles. The program starts by asking the user to select a shape and, depending on the selection, to input the necessary lengths. Then, it prints the area and terminates.

```
Select one of the following shapes by entering the corresponding number:
1 circle
2 rectangle
3 square
4 triangle

2
Enter the width of the rectangle: 4.0
Enter the height of the rectangle: 2.0

The area is 8.0.
```

2 Recursion

1. Define a function print_down_triangle(n) that prints a downside "right triangle" with base and height n and made of asterisks like the one below.

```
>>> print_down_triangle(5)
*****
****
***
**
*
```

2. Define a function print_up_triangle(n) that prints an upside "right triangle" with base and height n and made of asterisks like the one below.

```
>>> print_up_triangle(5)
*
**
***
****
*****
```

3. Generalise the function print_up_triangle(n) by defining a function print_up(print_line,n) that takes a function print_line(m) for printing the m-th line and a number of lines n and calls print_line starting from 1 up to n.

```
>>> def line_of_plusses(n):
  print('+' * n)
>>> print_up(line_of_plusses, 5)
+
++
+++
++++
+++++
```

4. Write a function `print_iso_triangle(n)` that prints an upside isosceles triangle made of asterisks like the one below. (Hint: use an auxiliary function).

```
   *
  ***
 *****
*******
```

5. Define a function `factorial(n)` that returns $n!$, the factorial of n ($n! = 1 \cdot 2 \cdot \ldots \cdot n$) using the algorithm:

$$n! = \begin{cases} 1 & \text{if } n \leq 1 \\ n \cdot (n-1)! & \text{otherwise} \end{cases}$$

6. Define a function `double_factorial(n)` that returns $n!!$ ($n!! = 1 \cdot 3 \cdot 5 \cdot \ldots \cdot n$ if n is odd and $n!! = 2 \cdot 4 \cdot 6 \cdot \ldots \cdot n$ if n is even).

7. Define a function `gcd(m,n)` that returns the greatest common divisor of `m` and `n` using Euclides' algorithm:

$$gcd(m,n) = \begin{cases} m & \text{if } m = n \\ gcd(m, n-m) & \text{if } m < n \\ gcd(m-n, n) & \text{if } m > n \end{cases}$$

8. Define a function `lcm(m,n)` that returns the least common multiple of `m` and `n`.

9. Define a function `sum_between(m,n)` that returns the sum of all integer numbers greater than `m` and smaller than `n`.

10. Define a function `sum_even_between(m,n)` that returns the sum of all integer even numbers greater than `m` and smaller than `n`.

11. Define a function `sum_odds_between(m,n)` that returns the sum of all integer odd numbers greater than `m` and smaller than `n`.

12. Define a function `is_prime(n)` that given a positive integer `n` returns `True` if `n` is prime and `False` otherwise. (Hint: use an auxiliary function).

13. Define a function `input_positive(message)` returns a positive integer by asking the user for input (displaying `message`) until a positive integer is provided or a malformed input causes an error.