# 4TPM205U: Array Algorithms : Sheet 2

## Iterate using a `while` loop

## Travaux dirigés

**Exercice 1** – ***(Euclidean Division )***

Let $a$ be a non-negative integer ($a \geq 0$) and $b$ is a positive integer ($b > 0$). Perform the euclidean division (integer division) of $a$ by $b$, to détermine the unique tuple of integers $(q, r)$ such that $a = bq + r$ and $0 \leq r < b$.

1. Write an algorithm of euclidean division by successive substractions. Run the algorithm for $a = 13$ and $b = 3$ ; for $a = 2$ and $b = 7$.
2. Determine, as a function of $a$ and $b$, the number of basic operations performed (comparisons, additions, subtractions).
3. Write two fonctions `reste(a,b)` and `quotient(a,b)` which return respectively the remainder and the quotient of the integer division of `a` by `b`.

**Exercice 2**

Keeping in mind, in a radix (or base) $b$ :

– the ***remainder*** of the division of $n$ by $b$ is equal to the value of the last digit of $n$ (the digit on the right most of $n$) ;

– the ***quotient*** of the integer division of $n$ by $b$ is equal to the value represented by the number $n$ without its last digit.

Write a function `sommeChiffres` which returns the sum of decimal digits of an integer $n$ passed as an argument.

**Exercice 3** – Given the following function :

```
def mystere (n, x):
    while (n // x > x and x != 0) :
        n = n // 10
        x = x - 1
    return n
```

What is the value returned by `mystere(100,6)` ? and by `mystere(1000,3)` ?

Correct the condition of iteration so that the function never generates an error.

**Exercice 4 – *(Rolling a 6-face dice)***

Let us assume that we have a function `lancerDe()` which simulates rolling a 6-face dice : the call to this function returns an integer between 1 and 6 chosen in a uniform random fashion.

1. Write a function `nbLancers6()` which returns the number of times the dice has to be rolled to obtain a 6. If the dice rolls a 1, then a 5, a 2, and then a 6, the function should return the value 4.

2. We roll now two dices at a time. Write a function `nbLancersDouble()` which returns the number of times the dice has to be rolled to obtain a double (*i.e.*, when both the dices roll the same value).

3. Write a function `moyenneTentativesDouble(n)` which calculates the average number of attempts to get a double, taken on a sample of `n` doubles obtained. For example, if `n` is 10, we roll the dices to obtain a double 10 times and we calculate the average over the number of attempts for each of these 10 trials.

4. Write a function `nbLancersPourSomme(n, value)` which calculates the number of times `n` dices are rolled to obtain a sum of the dices equal to `value`. What happens if the `value` passed as the argument of the function is not in the domain of possible values of `n` dices ?

**Exercice 5 – *(Racine carrée entière)***

The sum of $n$ first odd natural numbers is equal to the square of $n$.

For example, $1+3+5=3^2$. Another example, $1+3+5+7=4^2$, etc.

Deducing an algorithm to calculate the square root of a given positive integer ; calculate the number of additions and the comparisons performed.