- Contrôle sur les bases de l'algorithmique -

Exercice n°1 : On donne le programme suivant :

```
1  # programme 1
2  n = int(input("Donner un entier :"))
3  somme = 0
4  for k in range(n):
5      if k % 2 != 0:
6          somme = somme + k
7  print(somme)
```

1) Qu'affiche ce programme pour n = 10 ?

2) Quelle somme calcule ce programme ?

3) On souhaite savoir pour quelle valeur de l'entier n cette somme dépasse un seuil A donné par l'utilisateur. Écrire l'algorithme correspondant en langage naturel.

Exercice n°2 : On donne l'algorithme suivant :

```
n est un entier supérieur ou égal à 3
a ← 1
b ← 2
pour k allant de 3 à n faire :
        c ← a
        a ← b
        b ← c + a
afficher b
```

1) Qu'affiche cet algorithme pour n = 5 ?

2) Que fait cet algorithme à chaque passage dans la boucle ?

3) Écrire le programme correspondant en langage Python.

Exercice n°3 :

On lance deux dés équilibrés.
On souhaite savoir combien de lancers seront nécessaires avant d'obtenir un double.

1) Écrire l'algorithme correspondant en langage naturel.

On souhaite obtenir la probabilité d'obtenir un double.
Pour cela, on recommence un nombre n de fois l'expérience précédente, puis on affiche le pourcentage de réussite.

2) Écrire l'algorithme correspondant en langage naturel.

3) En utilisant la fonction Python **randint(1, 6)**, écrire la version Python de ce deuxième algorithme.

Exercice n°4 :

n est un entier choisi au hasard dans un intervalle $[a, b]$ (a et b sont aussi deux entiers).
On souhaite déterminer le nombre n en utilisant l'**algorithme de dichotomi**e.

1) Écrire l'algorithme correspondant en langage naturel.

2) Combien d'étapes maximales seront nécessaires si l'on choisit a = 0 et b = 32 ?

3) Écrire le programme correspondant en langage Python.

- Correction -

Exercice n°1 :

1) Pour n = 10, ce programme calcule la somme 1 + 3 + 5 + 7 + 9, et affiche donc **25**.

2) Ce programme calcule donc **la somme des entiers impairs de 1 à n - 1**.

3) Algorithme de seuil :

```
A est un nombre réel (le seuil à dépasser)
somme ← 0
n ← 0
Tant que somme < A faire :
        n ← n + 1
        Si n est impair faire :
                somme ← somme + n
Afficher n
```

Exercice n°2 :

1) Pour n = 5, la variable k va prendre les valeurs de 3 à 5 inclus.
On peut alors faire un tableau pour voir évoluer les variables au fil des boucles :

k	a	b	c
initialisation	**1**	**2**	
3	**2**	1 + 2 = **3**	1
4	**3**	2 + 3 = **5**	2
5	**5**	3 + 5 = **8**	3

Cet algorithme affichera donc la valeur **8** en choisissant n = 5 au départ.

2) A chaque passage dans la boucle, cet algorithme change les valeurs des variables a et b, en leur donnant respectivement les anciennes valeurs de b et a + b.

3) Programme correspondant en langage Python :

```
n = int(input("donner un entier:"))
a, b = 1, 2
for k in range(3, n + 1):
    a, b = b, a + b
print(b)
```

Exercice n°3 :

1) Algorithme pour afficher le nombre d'essais nécessaires pour faire un double :

```
de1 et de2 ← entier aléatoire entre 1 et 6
nb_essais ← 1
Tant que de1 ≠ de2 faire :
        nb_essais ← nb_essais + 1
        de1 et de2 ← entier aléatoire entre 1 et 6
Afficher nb_essais
```

2) Algorithme affichant la fréquence des doubles obtenus :

```
n est le nombre de lancers
succes ← 0
Pour k allant de 1 à n (ou de 0 à n - 1) faire :
        de1 et de2 sont les valeurs des deux dés
        Si de1 = de2 faire :
                succes ← succes + 1
frequence ← (succes * 100) / n
Afficher frequence
```

3) Programme Python correspondant :

```
from random import randint
n = int(input("Donner le nombre de lancers:"))
succes = 0
for k in range(n):
    de1 = randint(1, 6)
    de2 = randint(1, 6)
    if de1 == de2:
        succes = succes + 1
frequence = succes * 100 / n
print(frequence)
```

Exercice n°4 :

1) Algorithme en langage naturel :

```
n est un nombre entier de l'intervalle [a, b]
milieu ← quotient entier de (a + b) / 2
Tant que milieu ≠ n faire :
        Si milieu < n faire :
                a ← milieu
        Sinon faire :
                b ← milieu
        milieu ← quotient entier de (a + b) / 2
Afficher milieu
```

2) Si l'on choisit a = 0 et b = 32, comme $32 = 2^5$, 32 n'est divisible que 5 fois par 2.
Donc dans le pire des cas (le nombre à trouver est 0) , il faudra **6 étapes**.

3) Programme Python correspondant :

```
1  from random import randint
2  a, b = 0, 32                    # pour l'exemple
3  nombre = randint(a, b)
4  print(nombre)                   # pour vérifier la réponse
5  milieu = (a + b) // 2
6  while milieu != nombre:
7      if milieu < nombre:
8          a = milieu
9      else:
10         b = milieu
11     milieu = (a + b) // 2
12 print(milieu)
```

Remarque : Ce programme comporte un bug.
Seriez-vous le trouver ?
Et proposer une correction pour un bonus ?