Concours blanc - TP d'informatique - Sujet 1 - Correction

1. (a) *2 points*

```python
from math import *

def suite(n,u0):
    u = u0
    for k in range(n):
        u = cos(u)
    return u
```

(b) *3 points*

```python
from math import *
import matplotlib.pyplot as plt

def graphique(n,u0):
    u = u0
    X = [0]
    Y = [u]
    for k in range(n):
        u = cos(u)
        X.append(k+1)
        Y.append(u)
    plt.plot(X,Y,'x')
    plt.show()
```

2. (a) *2 points*

```python
def moyenne_sans_effectifs(X):
    s = 0
    for x in X :
        s += x
    return s / len(X)
```

(b) *3 points*

```python
def moyenne_avec_effectifs(X,N):
    s = 0
    n = len(X)
    effectif = 0
    for k in range(n):
        s += X[k]*N[k]
        effectif += N[k]
    return s/effectif
```

3. *3 points*

```python
def tri_insertion_avec_copie(L) :
    R = L.copy()
    n = len(R)
    for i in range(1,n) :
        x = R[i]
        j = i
        while j>0 and R[j-1] > x :
            R[j] = R[j-1]
            j = j-1
        R[j] = x
    return R
```

4. (a) *2 points*

   On définit la fonction f par :

$$f : \begin{array}{ccc} [0,1] & \to & \mathbb{R} \\ t & \mapsto & \dfrac{1}{1+t^3} \end{array}$$

   Tout d'abord, sur $[0,1]$, $1+t^3$ ne s'annule pas donc f est correctement définie.
   De plus, par les théorèmes opératoires, f est continue sur $[0,1]$.
   Ceci suffit à prouver que I est bien définie.

   (b) *2 points*

   $$I = \lim_{n\to+\infty} \frac{1}{n} \sum_{k=0}^{n-1} \frac{1}{1+\left(\frac{k}{n}\right)^3}$$

   (c) *3 points*

```python
def estimation(n=1000):
    s = 0
    for k in range(n):
        s += 1/(1+(k/n)**3)
    return (1/n)*s
```

Concours blanc - TP d'informatique - Sujet 2 - Correction

1. *3 points*

```python
def extrema(L):
    m = L[0]
    M = L[0]
    for e in L:
        if e > M:
            M = e
        if e < m:
            m = e
    return (m,M)
```

2. *2 points*

```python
def occurences(L,x):
    compteur = 0
    for e in L:
        if e == x:
            compteur += 1
    return compteur
```

3. (a) *1 points*

   Soit $M \in \mathcal{M}_{np}(\mathbb{K})$ ($\mathbb{K}$ étant $\mathbb{R}$ ou $\mathbb{C}$ et $n, p \in \mathbb{N}^*$)

   M est symétrique si et seulement M est égale à sa transposée.

   (b) *3 points*

```python
import numpy as np

def symetrique(M):
    n = M.shape[0]
    p = M.shape[1]
    for i in range(n):
        for j in range(p):
            if M[i,j]!=M[j,i]:
                return False
    return True
```

4. *3 points*

```python
def tri_selection_avec_copie(L):
    R = L.copy()
    n = len(R)
    for i in range(0,n-1):
        mini = R[i]
        indice_mini = i
        for j in range(i+1,n):
            if R[j] < mini:
                mini = R[j]
                indice_mini = j
        if indice_mini != i:
            R[i] , R[indice_mini] = R[indice_mini] , R[i]
    return R
```

5. (a) *2 points*

On définit la fonction f par :

$$f : \begin{array}{ccc} \mathbb{R} & \to &]0, +\infty[\\ x & \mapsto & e^x \end{array}$$

f est continue et strictement croissante sur $\mathbb{R}$.
Donc, par le théorème de la bijection, f est bijective de $\mathbb{R}$ dans $f(\mathbb{R}) =]0, +\infty[$.
Dans (E) admet une unique solution dans $\mathbb{R}$.
De plus, $e^0 \leqslant 3 \leqslant e^2$.
Donc, par croissance de f, $0 \leqslant \alpha \leqslant 2$.

(b) *3 points*

```python
from math import *
import matplotlib.pyplot as plt

def graphique():
    X = np.linspace(0,2,1000)
    Y_exp = [exp(x) for x in X]
    Y_droite = [3 for x in X]
    plt.plot(X,Y_exp,'b')
    plt.plot(X,Y_droite,'r')
    plt.show()
```

(c) *3 points*

```python
from math import *

def dichotomie(e):
    a = 0
    b = 2
    while b-a > e:
        m = (a+b)/2
        if exp(m)>3:
            b = m
        else:
            a = m
    return (a,b)
```

Concours blanc - TP d'informatique - Sujet 3 - Correction

1. (a) *2 points*

```
from math import *

def suite_liste(n):
    return [cos(k) for k in range(0,n+1)]
```

(b) *3 points*

```
from math import *

def min_suite(n):
    m = cos(0)
    rang = 0
    for k in range(1,n+1):
        if cos(k) < m :
            m = cos(k)
            rang = k
    return (m,rang)
```

(c) *2 points*

```
from math import *
import matplotlib.pyplot as plt

def graphique(n):
    X = [k for k in range(0,n+1)]
    Y = [cos(k) for k in range(0,n+1)]
    plt.plot(X,Y,'x')
    plt.show()
```

2. (a) *2 points*

La variance est donnée par : $s_x^2 = \overline{x^2} - \overline{x}^2 = \frac{1}{n}\sum_{k=1}^{n} x_k^2 - \left(\frac{1}{n}\sum_{k=1}^{n} x_k\right)^2$

(b) *3 points*

```
def variance(X):
    somme = 0
    somme_carre = 0
    n = len(X)
    for k in range(n):
        somme += X[k]
        somme_carre += X[k]**2
    return (1/n)*somme_carre - ((1/n)*somme)**2
```

(c) *1 point*

```
from math import *

def ecart_type(X):
    return sqrt(variance(X))
```

3. *2 points*

```
def recherche_mot_dans_chaine(mot, chaine):
    long_mot = len(mot)
    long_chaine = len(chaine)
    for i in range(long_chaine - long_mot +1) :
        j = 0
        while j < long_mot and chaine[i+j] == mot[j] :
            j+=1
        if j == long_mot :
            return True
    return False
```

4. (a) *1 point*

$$\mathrm{n}! = \prod_{\mathrm{k}=1}^{\mathrm{n}} \mathrm{k}$$

(b) *2 points*

```
def factorielle_non_rec(n):
    f = 1
    for k in range(1,n+1):
        f = f*k
    return f
```

(c) *2 points*

```
def factorielle_rec(n):
    if n == 0:
        return 1
    else:
        return factorielle_rec(n-1)*n
```