INFORMATIQUE : CONCOURS BLANC

CORRECTION

PARTIE A - PRÉLIMINAIRES

1.
```
def del_double(L):
    t = []
    a = L[0]
    t.append(a)
    for k in range(1,len(L)):
        if L[k] > a :
            a = L[k]
            t.append(a)
    return(t)
```
2. Pour le tableau `[3,1,5,3,2,8]` la fonction renvoie `[3,5,8]` le 1 et le 2, seuls, on été supprimés.
3.
```
def egal(L1,L2):
    if len(L1) != len(L2) :
        return ( False)
    for k in range(len(L1)):
        if L1[k]!=L2[k] :
            return(False)
    return(True)
```
4. Dans le pire des cas, les tableaux sont identiques, on fait une comparaison par passage dans le `for` donc un complexité de l'ordre de la taille de la liste.
5.
```
def checksum ( tab,chk) :
    S = 0
    for val in tab :
        S = S + abs(val)
    return ( S == chk)
```

PARTIE B - ANALYSEUR D'ATMOSPHÈRE

1.
```
def nb_echant(L):
    n = 0
    for val in L :
        if val == 1:
            n+=1
    return(n)
```
2.
```
def coherence(L)  :
    for val in L :
        if val != 1 and val != 0:
            return(False)
    return(True)
```
3. L'appel `descend([1,0,1,1,0,1],True)` doit renvoyer `[0,1,1,0,1,1]`
4.
```
def descend(L,b):
    for k in range(len(L)-1):
        L[k] = L[k+1]
    if b :
        L[-1] = 1  # derniere position
    else :
        L[-1] = 0
    return(L)
```
5.
```
def descend_cont(L,control,b):
    if not 0 in control : # le cas doit etre traité à part
```

```
        L[0] = 0
    for k in range(len(L)-1):
        if (not (k+1) in control) and L[k] == 0 : # le suivant n'est pas bloqué et il y a de la
            L[k] = L[k+1]
    if b :
        L[-1] = 1  # derniere position
    else :
        L[-1] = 0
    return(L)
```

6. `nb_echant(L)==0` ou `egal(L, [0]*len(L))`

PARTIE C - ANALYSE DES DONNÉES

1.
```
def lecture_fichier(fichier):
    f = open(fichier,mode='r')
    temps,oxygen,radiation = [ ],[ ]
    for ligne in f:
        if ligne[0] != '#':
            t,ox,yrad = ligne.split(':')
            temps.append(float(t))
            oxygen.append(float(ox))
            radiation.append(float(rad))
    f.close()
    return temps,oxygen,radiation
```

2.
```
radiatot = 0
for k in range(len(rad)-1):
    radiatot = radiatot + (temps[k+1]-temps[k]) * rad[k]
```

3.
```
var_oxy = 0
for k in range(len(oxy)-1):
    var_oxy = var_oxy + abs(temps[k+1]-temps[k])
```

4. `radiatot <= 10 and var_oxy < 20`

PARTIE D - ÉTUDE DE VIABILITÉ

Pour répondre a l'étude de viabilité il est nécessaire de résoudre une équation différentielle du second ordre de la forme $y" = f(y)$ dont les conditions initiales sont $y(0) = 10$ et $y'(0) = -0.02$
On considère que la fonction numérique f est saisie dans le code Python.

1. $\begin{cases} y' &= z \\ z' &= f(y) \end{cases}$ avec $y(0) = 10$ et $z(0) = y'(0) = -0,02$.

2.
```
n = 10000
h = 0.002
y,z = 10,-0.02 # valeurs initiales
Ly,Lz = [y],[z] #  listes de stockage
for k in range(n-1) :
    fy = f(y)
    y = y + z*h
    z = z + fy*h
    Ly.append(y)
    Lz.append(z)
```

3.
```
n = 10000
h = 0.002
y,z = 10,-0.02 # valeurs initiales
Ly,Lz = [y],[z] #  listes de stockage
for k in range(n-1) : # n points au total mais on a déjà le départ
    fy = f(y)
    y = y + z*h + h**2/2*fy
    z = z + h/2*(fy+f(y))
    Ly.append(y)
    Lz.append(z)
```