Informatique en CPGE (2018-2019)
Corrigé devoir surveillé 4

Exercice 1

`liste1` vaut `['Ceci','est','une','chaîne','de','caractères']`
`liste2` vaut `['19','08','2018']`

Exercice 2

```
ch=["Ceci","est","une","chaîne","de","caractères"]

f=open("fichier.txt","w")
for mot in ch:
    f.write(mot+"\n")
f.close()
```

Exercice 3

1. (a)

`[1,2,3]+[4,5,6]` vaut `[1,2,3,4,5,6]`

1. (b)

`2*[1,2,3]` vaut `[1,2,3,1,2,3]`

2.

```
def vsom(liste1, liste2):
    return [liste1[i] + liste2[i] for i in range(len[liste1)]

# ou bien
def vsom(liste1, liste2):
    s = []
    for i in range(len(liste1):
        s.append(lisle1[i] + liste2[i])
    return s
```

Exercice 4

1.

n passages dans la boucle externe et 10 passages dans la boucle interne
donc $n + n \times 10$ additions. Complexité linéaire de niveau $\mathcal{O}(n)$.

2.

n passages dans la boucle externe et n passages dans la boucle interne
donc $n + n \times n$ additions. Complexité quadratique de niveau $\mathcal{O}(n^2)$.

3.

n passages dans la boucle externe et i passages dans la boucle interne pour chaque i
donc $n + (1 + 2 + \ldots + (n-1) = n + n(n-1)/2$ additions.
Complexité quadratique de niveau $\mathcal{O}(n^2)$.

Exercice 5

```
def dicho(f, a, b, eps):
    while b - a > eps:
        m = (a + b) / 2
        if f(a) * f(m) > 0:
            a = m
        else:
            b = m
    return a, b
```

1. a et b constituent un encadrement de la solution à eps près.

2. L'amplitude initiale est $b-a=2$. Après k itérations, on obtient une amplitude de $\dfrac{b-a}{2^k}$ et on cherche k tel que $\dfrac{2}{2^k} \leq 10^{-2}$ soit : $2^k \geq 200$ et on obtient $k=8$.

3. L'amplitude initiale est $2,6-0,2=2,4$, et on cherche k tel que $2^k \geq 24$, donc $k=5$.

5 passages dans la boucle while donc 6 soustractions pour le test (5 positifs et le dernier négatif) et 5 additions, 5 divisions, 5 multiplications (soit 21 opérations), plus 5 fois le calcul de $f(a)$ et $f(m)$ qui nécessitent chacun 4 opérations ; on obtient donc 61 opérations.

(1.4 1.475)

Exercice 6

1.

```
def df(f, x):
    return (f(x + 10 ** (-5)) - f(x - 10 ** (-5))) / 2 / 10 ** (-5)
```

2.

```
def newton1(f, a, df, eps):
    x = a
    while abs(f(x)) > eps:
        x = x - f(x) / df(f, x)
    return x
```

3.

```
def newton2(f, a, df, eps):
    x = a
    u = x - f(x) / df(f, x)
    while abs(u - x) > eps:
        x = u
        u = x - f(x) / df(f, x)
    return u
```