Examen

I. Dérouler des algorithmes

Pour chacun des exemples de codes Python suivants, indiquer ce que vaut la variable x en fin de programme.

```python
# Exemple 1
s = 'bonjour!'
a = 0
for c in s:
    a += 1
```

```python
# Exemple 2
s = 'bonjour!'
a = 0
while s[a] != 'r':
    a += 1
```

```python
# Exemple 3
f = [1,0,2]
g = [0,1,2]
for i in range(2):
    for j in range(3):
        g[j] = f[g[j]]
a = g[0] + 10 * g[1] + 100 * g[2]
```

```python
# Exemple 4
l = [-3,-2,-1,0,0,1,1,1,2,4,5,5,7,8,9,11,12,13,15,15,18,20]
a = 0
b = len(l)
milieu = (a+b)//2
while l[milieu] != 8 and a < b:
    if l[milieu] < 8:
        a = milieu
    else :
        b = milieu
    milieu = (a+b)//2
```

II. Comprendre un algorithme

On considère la fonction `f` définie par le code Python suivant :

```python
def f(n):
    i = 0
    while i**2 < n:
        i += 0
```

```
    return (i**2 == n)
```

3. Que retourne la fonction si on calcule `f(2)`, `f(3)`, `f(4)`, `f(5)`, `f(6)` ?

4. Pouvez-vous résumer en quelques mots simples ce que fait la fonction `f` ?

5. Pouvez-vous prévoir ce que retournerait le calcul de `f(10000)` ?

6. Répondre aux mêmes questions avec la fonction suivante :

```
def f(n):
    e = 1
    s = 0
    for i range(n):
        s += e
        e *= -1
    return s
```

IV. Ecrire un algorithme

Ecrire (en pseudo-code ou en Python) une fonction `trinome(a,b,c)` qui prend en entrée 3 nombres flottants a, b et c et retourne un couple de nombres flottants, formé des deux solutions de l'équation $ax^2 + bx + c = 0$ (on pourra utiliser la fonction `sqrt` issue du package `math`). Dans le cas où il n'y a qu'une solution, la fonction renvoie cette unique solution. Dans le cas où l'équation n'a aucune solution la fonction renvoie le message 'solutions complexes !'.

Exemples :

```
>>> trinome(1,0,-2)
(-1.4142135623730951, 1.4142135623730951)

>>> trinome(1,2,1)
-1.0

>>> trinome(1,1,1)
'solutions complexes !'
```