Série N° 2

Exercice 1

On considère l'algorithme suivant qui consiste à recherche un élément dans une liste :

```
def ChercherElement(L,a) :
        i=0
        while i < n and  L[i]== a :
                i=i+1
        if i<n :
                return True
        else :
                return False
```

On considère L = [5, 1, 2, 4, 6, 7].

1) Quel est le nombre d'opérations effectuées par l'algorithme de recherche avec les entrées L et 5 ?
2) Quel est le nombre d'opérations effectuées par l'algorithme de recherche avec les entrées L et 7 ?
3) Quel est le nombre d'opérations effectuées par l'algorithme de recherche avec les entrées L et 3 ?

Exercice 2

Évaluer le nombre d'exécutions pour les quatre programmes suivants :

Programme1 :
```
for i in range(n):
   print("pif")
for j in range(n):
    print("paf")
for k in range(n):
    print("pouf")
```

Programme 2 :
```
for i in range(n):
    print("pif")
    for j in range(n):
          print("paf")
for k in range(n):
    print("pouf")
```

Programme 3 :
```
for i in range(n):
   print("pif")
   for j in range(n):
       print("paf")
       for k in range(n):
            print("pouf")
```

Programme 4 :
```
for i in range(1, 1+n):
   for j in range(i):
      print(i+j)
```

Exercice 3

On définit la suite $(x_n)\ n \in N$ par x0 = 1 et $\forall n \in N$, $x_{n+1} = \sin(x_n)$.
On désire écrire une fonction **iteres(n)** dont la valeur de retour est la liste [x0 , . . . , xn]

Les étudiants **Anastragon, Bector et Coryphime** proposent respectivement les codes suivants :

Code d'Anastragon :	**Code de Bector :**	**Code de Coryphime:**

def iteres (n) :
  res = np.arange(n+1)+1.
  while n > 0 :
  res = np.sin(res)
  n =n - 1
  return (res)
	def iteres (n) :
	  res = (n+1)*[1.]
	  for i in xrange (n) :
	  res[i+1]=sin(res[i])
	  return (res)
		def x (n) :
		  res = 1.
		  while n > 0 :
		  res = sin (res)
		  n =n - 1
		  return (res)
		def iteres (n) :
		  res = range (n+1)
		  for i in res :
		  res[i] = x(i)
		  return (res)

1) Parmi ces 3 codes, lequel est correct et a une complexité de l'ordre de O(n2) ?
2) Lequel est correct et a une complexité de l'ordre de O(n) ? Lequel est faux ?
3) Le code faux s'exécute–t–il ? Si oui que calcule–t–il ?

Exercice 4

Considérer 7 algorithmes A0, A1, ..., A6 conçus pour résoudre un même problème de taille n. La complexité en temps de chacun de ces algorithmes est exprimée dans la table ci-dessous.

Algorithme	Complexité	Temps		
		n=10	n=100	n=1000
A0	O(1)			
A1	$O(\log_2(n)$			
A2	$O(\sqrt{n})$			
A3	O(n)			
A4	$O(n^2)$			
A5	$O(n^3)$			
A6	$O(2^n)$			

1) Calculer les ordres de grandeurs suivantes en secondes : 1 heure, 1 jour, 1 semaine, 1 mois , 1 année, 1 siècle et 1 millénaire

2) En supposant qu'une unité de taille s'exécute en une milliseconde, calculer le temps nécessaire pour traiter des tailles respectivement égales à 10, 100 et 1000.
3) Répéter la question 2 avec une unité de temps égale à une microseconde
4) Que peut-on en conclure ?