TYPE EXAMEN FINAL

1) Quel est le résultat de l'expression: 5%2

(a) 0 (b) 1 (c) 2 (d) 3 (e) 4

2) Quels programmes contiennent un appel de fonction?

(I) "BONJOUR".lower()
(II) def maFonction(x):
return x
(III) print("Bonjour")

(a) (I), (II), et (III) (b) (I), et (II) (c) (I) et (III)
(d) (II) et (III) (e) (I) seulement

3) Qu'est-ce que le programme Python suivant va afficher?

```
s = "abcdefgh"
x = "a"
while x in s:
        s = s[ : len(s)-1]
        print(x, end = " ")
```

(a) Rien. (b) Erreur (c) a b c d e f g h
(d) a a a a a a a a (e) b c d e f g h

4) Que réalise la fonction suivante ?

```
def maFonction(L):
   p = 0
   for i in range(1, len(L)):
      if L[i] >= L[p]:
         p = i
   return p
```

(a) Retourne la valeur entière maximale de la liste L
(b) Retourne la valeur entière minimale de la liste L
(c) Retourne la position (i.e. indice) de la valeur entière minimale de la liste L, s'il y'en a plusieurs, elle retoure la position de la première valeur minimale
(d) Retourne la position (i.e. indice) de la valeur entière maximale de la liste L, s'il y'en a plusieurs, elle retoure la position de la première valeur maximale
(e) Retourne la position (i.e. indice) de la valeur entière maximale de la liste L, s'il y'en a plusieurs, elle retoure la position de la dernière valeur maximale

5) Qu'est-ce que le programme Python suivant va afficher?

```
class maClasseA:
    def __init__(self):
        self.x = 0
        self.y = 0

class maClasseB(maClasseA):
    def affiche(self):
        print(self.x,self.y)

obj = maClasseB()
obj.affiche()
```

(a) Rien
(b) 0 0
(c) 1 1
(d) Rien. Message d'erreur à cause de l'appel incorrect du constructeur dans la ligne 7, (i.e. : TypeError: __init__() missing 1 required positional argument).
(e) Rien. . Message d'erreur à cause que les objets du type *maClasse*B n'ont pas d'attributs (i.e. variable d'instance) x, (i.e.: AttributeError: '*maClasse*B' object has no attribute 'x').

6) La suite Fibonacci, est une suite de nombres dont chaque terme est égal à la somme des deux termes qui le précèdent.
Par exemple, les 10 premiers nombres Fibonacci sont:
0 1 1 2 3 5 8 13 21 34 55 89
Lequel des énoncés suivants est correct à propos de la fonction suivante :

```
def maListeFib(n):
        '''(int)->list
        Précondition: n n'est pas négatif
        Retourne une liste contenant les n premiers termes de la suite Fibonacci'''.
        if n == 0:
                return [0]
        elif n == 1:
                return [0, 1]
        l = [0, 1]
        for i in range(2, n+1):
                l.append(l[i-1] + l[i-2])
        return l
```

(a) La fonction est correcte. Elle renvoie une liste contenant les n premiers termes de la suite Fibonacci.
(b) La fonction est incorrect. Elle peut être corrigé en remplaçant la dernière ligne avec : `return maListeFib(n-1)`
(c) La fonction est incorrect. Elle peut être corrigé en remplaçant for `i in range(2, n+1):` par `for i in range(2, n+1,2):`
(d) La fonction est incorrect. Elle peut être corrigé en remplaçant la dernière ligne par : `return [len(l)-1]`
(e) La fonction est erronée parce qu'elle n'utilise pas la récursivité.

7) Qu'est-ce que le programme Python suivant va afficher?
Le programme utilise la fonction de tri Python ***sort.***

```
ages = [22, 19, 23, 10, 34, 31, 20]
ages.sort()
print(ages[0])
```

(a) 10
(b) 22
(c) 34
(d) Aucune réponse n'est valide
(e) Message d'erreur

8) Supposons qu'une classe `chose` a une méthode appelée *faire*. Le reste du corps de cette méthode et les autres méthodes de la classe sont omis ci-dessous pour économiser de l'espace.

```
class chose():
    def faire(self, a, b):
        ...
```

Supposons que *toto* est une variable qui fait référence à un objet de type `chose` et que *da1* et *da2* sont deux autres variables qui ont été initialisées à une certaine valeur. Comment doit-on appeler la la méthode *faire* ?

(a) toto.faire(da1,da2)
(b) chose.faire(da1,da2)
(c) self.faire(toto, da1,da2)
(d) faire(toto,da1,da2)

9) Sachant que X, Y est Z sont des variables booléennes, on veut dériver une expression booléenne qui est évaluée à la valeur `False` si une ou plusieurs des trois variables sont fausses et `True` dans le cas contraire. Lesquelles des expressions suivantes satisfont cette définition ?

(I) *X and Y and Z*
(II) *(not X) or (not Y) or (not Z)*
(III) *(not X) and (not Y) and (not Z)*

(a) Uniquement I (b) Uniquement II; (c) Uniquement III
(d) II et III (e) I, II et III

10) Dans la fonction suivante combien d'arguments (ou paramètres actuels) y-a-t'il dans l'appel `maFonction(lst)`?

```
def maFonction(x):
    i=0
    while i < len(x):
        print(x[i], end=" ")
        i=i+1

lst=[1,12,3,5,8,7]
maFonction(lst)
```

(a) 0 (b)1 (c) 2 (d) 6 (e) Aucune réponse n'est valide

11) Laquele des instructions suivante(s) sont équivalentes au morceau du code suivant ?

```
if len(s) < 5 or "x" in s:
        return False
else:
        return True
```

I) return len(s) <5 or "x" in s
II) return not(len(s) <5 or "x" in s)
III) return len(s) >=5 and "x" not in s

(a) Uniquement I (b) Uniquement II (c) Uniquement III (d) I et III
(e) II et III

12) Combien de lettres A le code suivant va t-il afficher ?

```
def monA(n):
    for i in range(n):
        if i % 8 == 0:
            print("A" *i)

monA(11)
```

(a) 2 (b) 3 (c) 8 (d) 26 (e) 27

13) Pour la fonction suivante, combien d'opérations sont exécutées dans la fonction, approximativement, quand n devient large (n est le nombre d'éléments dans la liste L)?

```
def monbigO(lst):
        '''(list de int)->None"
        n=len(lst)
        for i in range(n):
                for j in range(20):
                        lst[i] = lst[i] + 1
        for i in range(n):
                print(lst[i])
```

(a) $O(\log_2 n)$ (b) $O(n)$ (c) $O(n \log_2 n)$ (d) $O(n^2)$ (e) $O(n^3)$

14) Qu'est-ce que le programme Python suivant va afficher?

```
a = [10, 20, 30, 40]
b = a
a[1] = 0
print(b)
```

(a) [10, 20, 30, 40] (b) [0, 20, 30, 40] (c) [10, 0, 30, 40]
(d) [10, 20, 0, 40] (e) [10, 20, 40, 0]

15) Qu'est-ce que le programme Python suivant va afficher?

```
a = [10, 20, 30, 40]
b = a[:]
a[1] = 0
print(b)
```

(a) [10, 20, 30, 40] (b) [0, 20, 30, 40] (c) [10, 0, 30, 40]
(d) [10, 20, 0, 40] (e) [10, 20, 40, 0]

16) Qu'est-ce que le programme Python suivant va afficher ?

```
class Point:
        def __init__(self, x=0, y=0):
                self.x = x
                self.y=y
        def __repr__(self):
                return 'Point('+str(self.x)+','+str(self.y)+')'

def f(x,a,b):
        a=b
        a.x=1
        a.y=1
        x=x+2

x=0
p1=Point(1,2)
p2=Point(3,4)
f(x,p1,p2)
print(x, p1, p2)
```

(a) 0 Point(1,2) Point(1,1)
(b) 0 Point(1,1) Point(3,4)
(c) 0 Point(1,2) Point(3,4)
(d) 2 Point(1,2) Point(3,4)
(e) 2 Point(1,1) Point(3,4)

17) Quelle serait la meilleure description de la fonction suivante ?

```
def maFonction(n):
   res=1
   for i in range(n):
      res=res*n
   return res
```

(a) Elle retourne n^n
(b) Elle retourne factorial n
(c) Elle retourne n*n
(d) Elle retourne n
(e) Elle retourne 1
(f) Elle crache pour toute valeur $n \geq 1$

18) Que fait la fonction Python suivante? (si la liste L des nombres entiers n'est pas vide)

```
def maFonc(L,k):
    if len(L)==0:
        return True
    return L[0]>k and maFonc(L[1:],k)
```

(a) Retourne True si tous les éléments de L sont supérieurs à k, et False sinon.
(b) Retourne True si L est vide et False sinon.
(c) Retourne True si au moins un des éléments de L est supérieur à k, et False sinon.
(d) Retourne True si le premier éléments de L est supérieur à k, et False sinon.
(e) Exécute une boucle infinie.

19) Une chaîne de caractères s1 est un anagramme de la chaîne s2 si ses lettres peuvent être réarrangées pour formuler s2. Par exemple, " listen" est un anagramme de "silent", et "admirer" est un anagramme de "married". Examiner ce code :

```
def est_anagram (s1, S2) :
        ''' (str, str) - > bool
        renvoie True si et seulement si s1 est un anagramme de s2.
        Précondition : caractères dans s1 et S2 sont minuscules de l'alphabet Anglais

         > > > est_anagram("ab", "a")
        False
         > > > est_anagram("silent", "listen")
        True
         > > > est_anagram("today", "today")
        True
        '''
```

Lequel des algorithmes suivant(s) c.-à-d solution(s) peut être utilisé pour implémenter la function `est_anagram`?

(I)
Pas 1. Transformez s1 en une liste de caractères L1. (e.g., si s1='abba', L1=['a','b','b','a']
Pas 2. Transformez s2 en une liste de caractères L2.
Pas 3. Pour chaque élément de L1, éliminez une occurrence de cet élément de L2 (s'il existe).
Pas 4. Si L2 devient vide, s1 est un anagramme de s2.

(II)
Pas 1. Transformez s1 en une liste de caractères L1.
Pas 2. Transformez s2 en une liste de caractères L2.

Pas 3. Triez les deux listes. (trier une liste de caractères consiste à ordonner les caractères en ordre alphabétique)
Pas 4. Si L1 == L2, s1 est un anagramme de s2.

(a) I seulement (b) II seulement
(c) I et II (d) Aucune des d_eclarations ci-dessus.

20) *Que-ce que le programme Python suivant va afficher sur l'écran?*

```
def soustraction (a,b) :
   if ( a >= b ) :
      return (soustraction (a-b, b))
   else :
      return a

diviseur=20
dividende=3
Resultat=soustraction(diviseur,dividende)
print ("Resultat : " + str (Resultat))
```

(a) `Resultat: 2`
(b) `Resultat: 0`
(c) `Resultat: 17`
(d) Erreur

21) *Que-ce que le programme Python suivant va afficher sur l'écran?*

```
class CasNormal:
   def uneMethode(self):
      print("normal")
class CasSpecial:
   def uneMethode(self):
      print("special")
def casQuiConvient(estNormal=True):
   return CasNormal() if estNormal else CasSpecial()

uneInstance = casQuiConvient(estNormal=False)
uneInstance.uneMethode()
```

(a) normal
(b) special
(c) Erreur
(d) Aucune des réponses ci-dessus

22) Combien d'appels de la fonction `maFonction` sont effectués dans le programme suivant ?

```
def maFonction(n):
        if n > 0:
                print( n % 10)
                maFonction (n // 10)

maFonction(8540)
```

(a) 5
(b) 4
(c) 10
(d) infini
(e) Aucune des réponses ci-dessus

23) Qu'est-ce que le programme Python suivant va afficher?

```
class monNom:
   def __init__(self, a, b, c):
      self.prenom = a
      self.moi = b
      self.nom = c

prenom = "Nadine"
nom = monNom(prenom, 'F', "Smith")
prenom = "Paul"
nom.nom= "Le"
print(nom.prenom, nom.nom)
```

(a) Nadine Le
(b) Nadine F Smith
(c) Nadine Smith
(d) Erreur
(e) aucune de ces réponses

24) Une ligne du programme suivant manque :

```
for i in range(1, 7) :
        # ligne qui manque
                if j <= i:
                        print(j, end=" ")
                else:
                        print(" ", end=" ")
        print()
```

Quelle est la ligne qui manque pour que le programme affiche ce qui suit:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5
```

(a) for j in range(6, 0, -1):
(b) for j in range(1,6):
(c) for j in range(1,7):
(d) for j in range(7):
(e) for j in range(6):

25) Quelle est la description correcte de la fonction suivante, si L est une liste des entiers qui n'est pas vide?

```
def maFonction(L):
        X=sorted(L) # sorted(L) retourne une version triée de la liste L
        if len(X)==1:
                return True
        for i in range(len(X)):
                if i==0 and X[i]!=X[i+1]: # X[i] est le premier élément
                        return True
                elif i==len(L)-1 and X[i]!=X[i-1]:# X[i] est le dernier élément
                        return True
                elif i!=0 and i!=len(L)-1: # pas le premier ou le dernier
                        if X[i]!=X[i-1] and X[i]!=X[i+1]:
                                return True
        return False
```

(a) Retourne True si les élément de L sont tous différents (i.e., si L ne contient pas de doubles), et False sinon.
(b) Retourne False si les élément de L sont tous différents (i.e., si L ne contient pas de doubles), et True sinon.
(c) Retourne True si L contient au moins un élément qui arrive exactement une fois dans L, et False sinon.
(d) Retourne False si L contient au moins un élément qui arrive exactement une fois dans L, et False sinon.
(e) Aucune des d_eclarations ci-dessus.

26) Considérer la fonction suivante qui calcule n*x pour tout entier n ≥0:

```
def maSomme(x,n):
    '''Return x*n, fonctionne seulement pour les entiers non négatives de n'''

    if n <= 0: return 0
    return x + maSomme (x, n-1)
```

Si nous appelons cette fonction avec n = 26, alors :
(a) cette fonction effectue 25 appels de fonctions récursives et 25 additions
(b) cette fonction effectue 26 appels de fonctions récursives et 26 additions
(c) cette fonction effectue 27 appels de fonctions récursives et 26 additions
(d) aucune des réponses ci-dessus.

27) Une partie de la fonction suivante manque.

```
def maFonction(l):
    '''(list of int)->list of int
    Retourne une liste avec tous les entiers x de l pour lesquels -x est aussi dans l
    >>> neg([1, 1, 100, -1, -20, -300, 20, 55, 55])
    [1, 1, -1, -20, 20]
    >>> neg([-1, 100, 55, 55])
    []
    '''
    l2=[]
    # CODE MANQUANT
    return l2
```

Quel est le fragment de code qui manque?

(I)

```
for x in l:
    if -x in l:
        l2.append(x)
```

(II)

```
for x in l:
    if x<0 and x in l:
        l2.append(-x)
```

(III)

```
for x in l:
    pas_ajoute=True
    for y in l:
        if x+y==0 and pas_ajoute:
            l2.append(x)
            pas_ajoute=False
```

(a) I et II
(b) I et III
(c) I seulement
(d) II seulement
(e) III seulement