Université d'Ottawa
Faculté de Génie

École de Science
d'Informatique
et de Génie Électrique

University of Ottawa
Faculty of Engineering

School of Electrical
Engineering
and Computer Science

uOttawa

L'Université canadienne
Canada's university

ITI1520A Introduction à l'Informatique I

EXAMEN FINAL

Durée de l'Examen: 3 h **19 Décembre 2017, 19:00-22 :00**
Professeure : Zhor Sebbani **Page 1/ 24**

Nom et prénom de l'étudiant: _____

Numéro de l'étudiant: _____

Signature: _____

Note : / 56

INSTRUCTIONS:

1. Le nom et numéro de l'étudiant doivent être écrits sur la feuille de réponse scantron.
2. L'examen se déroule à livre fermé, aucun document n'est autorisé.
3. Aucune calculatrice n'est autorisée.
4. Vous devez répondre à toutes les questions (56). Vous cochez sur la feuille scantron le numéro de la bonne réponse (une seule bonne réponse par question).
5. Chaque question est évaluée à **1 point**.
6. Veuillez remettre le formulaire de réponses AINSI que l'examen.
7. Il est interdit de se servir de téléphones cellulaires, de dispositifs électroniques non autorisés ou de notes de cours.

1) Quel est le résultat de l'expression: 4%7

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

2) Quels programmes contiennent un appel de fonction?

(I) "melekeok".upper()
(II) round(3.21)
(III) aire(2, 9)
(IV) def mysquare(x,2):
*return x*x*
(V) class Point:
pass

(a) Tous (I), (II), (III), (IV) et (V)
(b) (I), (II), (III), et (IV)
(c) (I), (II), et (III)
(d) (III) uniquement
(e) (II) et (III)

3) Si *a* est un nombre entier de trois chiffres, quelles commandes produit le chiffre du milieu de *a* ? (si a est 924, le chiffre du milieu est 2)

(I) (a // 10) % 10 (II) (a % 100) //10 (III) (a % 10) //10

(a) (I) (b) (II) (c) (III) (d) (I) et (II) (e) Aucune réponse n'est valide

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

```
s = "abcdef"
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 (d) a a a a a a (e) b c d e f

5) Que réalise la fonction suivante ?

```
def hilo(L):
    '''(list de int)->int
    Précondition: L contient au moins un entier'''
    pm = 0
    for i in range(1, len(L)):
        if L[i] >= L[pm]:
            pm = i
    return pm
```

(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

6) Qu'est-ce que le morceau du programme Python suivant va afficher?

```
import math
def size_format2(b):
        units = ['B', 'KB', 'MB', 'GB', 'TB']
        u = int(min(math.floor(math.log(b, 1000)), 4))
        return str(b/(1000**u))+units[u]

print(size_format2(200028))
```

(a) 200.028KB (b) 200.0KB (c) 2.0KB (d) 2B (e) 228.KB

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

```
class A:
        def __init__(self):
                self.x = 1
class B(A):
        def display(self):
                print(self.x)
obj = B()
obj.display()
```

(a) Rien
(b) 0
(c) 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 B n'ont pas d'attributs (i.e. variable d'instance) x, (i.e.: AttributeError: 'B' object has no attribute 'x').

8) Le nième nombre F_n de la suite Fibonacci, est défini pour les entiers non-négatifs n, comme suit:
• le nombre 0 Fibonacci est 0, i.e $F_0 = 0$,
• le 1er nombre Fibonacci est 1, i.e. $F_1 = 1$, et
• le nième nombre Fibonacci F_n est défini récurssivement comme $F_n = F_{n-1} + F_{n-2}$.

Par exemple, les 20 premiers nombres Fibonacci sont:
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181.
Lequel des énoncés suivants est correct à propos de la fonction suivante :

```
def fib(n):
    '''(int)->int
    Précondition: n n'est pas négatif
    Retourne le nième nombre Fibonacci'''
    if n == 0:
        return 0
    elif n == 1:
        return 1
    l = [0, 1]
    for i in range(2, n+1):
        l.append(l[i-1] + l[i-2])
    return sum(l) #sum(l) retourne la somme des éléments de la liste l
```

(a) La fonction est correcte. Elle renvoie le n-ème nombre de Fibonacci.
(b) La fonction est incorrect. Elle peut être corrigé en remplaçant la dernière ligne par: `return fib(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 l[len(l)-1]`
(e) La fonction est incorrect. Elle peut être corrigé en remplaçant la dernière ligne par: `return fib(n-2)`

9) Dans le cadre de la programmation d'un jeu, un joueur a besoin d'être invité à entrer son choix comme une lettre A (s'il veut lutter contre le monstre bloquant son chemin) ou une lettre B (s'il veut trouver une autre route vers le trésor). Le jeu a besoin de poser cette question à plusieurs reprises jusqu'à ce qu'une lettre A ou B soit entré. Les deux versions minuscules et majuscules A, ou B sont acceptées comme valables de choix. On vous dit que vous pouvez supposer que le lecteur entre une lettre mais pas nécessairement A OU B.
Une ligne manque au morceau du code suivant. Quelle option parmi les suivantes résoudrait le problème de manière appropriée ?

```
choix=input("Entrer A pour combattre ou B pour trouver un autre itinéraire pour le
trésor: ")
# LIGNE DE CODE MANQUANT
    print(choix, "option non existante. SVP essayer à nouveau.")
    choix=input("Entrer A pour combattre ou B pour trouver un autre itinéraire pour
    le trésor: ")
```

(a) while choix.lower()!='a' or 'b':
(b) while choix.lower()!='a' or choix.lower()!='b':
(c) while choix.lower()!='a' and choix.lower()!='b':
(d) while choix.lower() in ['a','b']:
(e) while choix.lower()=='a' or choix.lower()=='b':

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

```
class Thing:
    def do_it(self, a):
        ...
```

Supposons que t est une variable qui fait référence à un objet de type `Thing` et que d est une autre variable qui a été initialisée à une certaine valeur. Comment doit-on appeler la la méthode *do_it* ?

(a) t.do_it(d) (b) Thing.do_it(d) (c) self.do_it(t, d)
(d) do_it(d) (e) do_it(t,d)

11) Sachant que A, B est C sont des variables booléennes, on veut dériver une expression booléenne qui est évaluée à la valeur True si une ou plusieurs des trois variables sont fausses et false dans le cas contraire. Lesquelles des expressions suivantes satisfont cette définition ?

(I) (not A) and (not B) and (not C)
(II) (not A) or (not B) or (not C)
(III) not (A and B and C)

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

12) Dans votre devoir 3, on vous a demandé d'implémenter la fonction appelée `is_rigorous` décrite ci-dessous. Il y a 4 lignes qui manquent au code. Laquelle des options suivantes résoud le problème correctement.

```
def is_rigorous(l):
    '''list de str->bool
    Retourne True si chaque élément de la liste apparaît exactement 2 fois ou
    la liste est vide. Sinon, elle renvoie False
    Précondition: Chaque élément de la liste apparaît un nombre pair de fois '''
    if len(l)==0: return True
    l=sorted(l)
    # 4 lignes manquantes au code
```

(a)
```
for i in range(len(l)-2):
    if l[i]==l[i+2]:
        return True
return False
```
(b)
```
for i in range(len(l)-2):
    if l[i]==l[i+2]:
        return False
return True
```
(c)
```
for i in range(len(l)-1):
    if l[i]==l[i+1]:
        return True
return False
```
(d)
```
for i in range(len(l)-1):
    if l[i]==l[i+1]:
        return False
return True
```
(e) Aucune réponse n'est valide

13) S*érie* est une séquence maximale de valeurs répétées consécutives. Maximale (signifie que vous ne pouvez pas le rendre plus long tout en étant toujours une séquence de valeurs répétées consécutives). Par exemple dans la liste suivante [1,1,2,0,0,0,4,5,0,0,5,5,5,5], 1,1, est une s*érie*, et sont donc aussi : 2, et 0,0,0 et 4, et 5 et 0,0, et 5,5,5,5.
Qu'est-ce que la fonction suivante ayant une liste x d'entiers comme entrée réalise?

```
def wahiawa(x):
    '''(list of int)->None'''
    i=0
    while i < len(x):
        print(x[i], end=" ")
        while (i<len(x)-1 and x[i]==0 and x[i+1]==0):
            i=i+1
        i=i+1
```

(a) Affiche tous les éléments de x sauf les zéros, c'est à dire qu'elle n'affiche pas les éléments à partir de séries de zéros.
(b) Affiche tous les éléments de x sauf que dans chaque série de zéros elle n'affiche que le premier zéro de la *série*.
(c) Affiche tous les éléments de x sauf que dans chaque série de zéros elle affiche seulement le premier et le deuxième zéro.
(d) Affiche tous les éléments de x jusqu'à ce qu'elle rencontre le premier zéro. Puis elle n'affiche rien d'autre après.
(e) Affiche quelque chose suivie d'un message d'erreur.

14) Dans la question précédente, combiens d'arguments (ou paramètres actuels) y-a-t'il dans l'appel `wahiawa`?

(a) 0 (b)1 (c) 2 (d) 12 (e) Aucune réponse n'est valide

15) Quelle est description la plus correcte de ce que fait la fonction suivante ?

```
def waipahu(x):
    '''(list)->bool
    Précondition: les éléments de x sont des nombres et x a au moins  2 éléments
    '''
    result=True
    for i in range(len(x)-1):
        if(x[i]>x[i+1]):
            result=False
        else:
            result=True
    return result
```

(a) La fonction renvoie True si les chiffres dans la liste x sont classés du plus petit au plus grand. Dans le cas contraire, elle renvoie False.
(b) La fonction renvoie True si les chiffres dans la liste x sont classés du plus grand au plus petit. Dans le cas contraire, elle renvoie False.
(c) La fonction retourne False si l'avant-dernier élément est plus grand que le dernier élément et True sinon.
(d) Aucune des réponses ci-dessus.

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

```
if len(s) < 2 and "?" in s:
    return False
else:
    return True
```

I) return len(s) <2 and "?" in s
II) return not(len(s) <2 and "?" in s)
III) return len(s) >=2 or "?" not in s

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

17) La fonction suivante permet de trier une liste a de nombres :

```
0: def bubble_sort(a):
1:     for i in range(len(a)):
2:         for j in range(len(a)-1):
3:             if a[j] > a[j+1]:
4:                 a[j], a[j+1] = a[j+1], a[j] # échange
```

Si la liste d'entrée, a, contient 10 000 éléments, la ligne 3 `de bubble_sort(a)` s'exécute un nombre de fois égal à :

(a) 9 900 (b) 10 000 (c) 99 900 (d) 100 000 (e) 99 990 000

18) Si la liste d'entrée, a, contient 10 000 éléments qui sont déjà triés du plus petit au plus grand, la ligne 4 de `bubble_sort(a)` s'exécute un nombre de fois égal à :

(a) 0 (b) 10 000 (c) 9 900 (d) 99 900 (e) 99 990 000

19) Combien d'étoiles le code suivant va t-il afficher ?

```
def ewa_beach(n):
    for i in range(n):
        if i % 8 == 0:
            print("*" * (i+1))
ewa_beach(19)
```

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

20) 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 nanakuli(L):
    '''(list de int)->None"
    n=len(L)
    for i in range(n):
        for j in range(25):
            L[i] = L[i] + 10
    for i in range(n):
        print(L[i])
```

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

21) Laquelle des lignes ci-dessous, qui une fois placée dans l'emplacement indiqué dans le code, résoud le problème décrit dans la description de la fonction correctement.

```
def encrypt(s):
    '''str ->str
    Renvoie une nouvelle chaîne de caractères où le premier et le dernier caractère
    de s deviennent le premier et deuxième dans la nouvelle chaîne, puis le deuxième et
    l'avant-dernier deviennent le 3ème et 4ème ...
    jusqu'à ce qu' il ne reste plus de lettres dans s.

    Précondition: len(L) >= 2 and len(L) % 2 == 0
    >>> encrypt('ab')
    'ab'
    >>> encrypt('1234abcd')
    '1d2c3b4a' '''

    snew=''
    for i in range(0,len(s)//2):
        # CODE MANQUANT ICI
    return snew
```

(a) snew=snew+s[i]+s[len(s)//2+i]
(b) snew=snew+s[i]+s[len(s)-i]
(c) snew=snew+s[i]+s[len(s)-i-1]
(d) snew=snew+s[i]+s[len(s)-i+1]
(e) snew=snew+s[i]+s[2*i]

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

```
def lahaina(x, y):
    print(x+y)

z = 10 * lahaina(5,5)
print(z)
```

(a) 250 (b) 10 (c) 100 (d) 10 et 100 (e) 10 et un message d'erreur.

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

```
a = [1, 2, 3, 4]
b = a
a[1] = 0
print(b)
```

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

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

```
a = [1, 2, 3, 4]
b = a[:]
a[1] = 0
print(b)
```

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

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

```
def Kaneohe(v, w):
        v=v*w[0]
        w[1]=w[0]+w[2]

c=100
d=[2.5, 0, 1]
Kaneohe(c,d)
print(c,d)
```

(a) 100 [2.5, 3.5, 1] (b) 100 [250.0, 3.5, 1] (c) 250.0 [2.5, 3.5, 1]
(d) 100 [2.5, 0, 1] (e) 250.0 [2.5, 0, 1]

26) Le corps de la fonction suivante manque.

```
def même(s1,s2) :
        '''(string,string)- >bool
        précondition: len(s1)==len(s2); len(s1) > 1
        Considérant deux chaînes s1 et S2 de même longueurs et de longueur au moins
        un, la fonction renvoie True si les deux chaînes sont les mêmes et False sinon.
        '''
```

Lequel des énoncés suivants correspond au corps de la fonction qui permettrait de résoudre correctement le problème?

(I)
```
for i in range(len(s1)):
        for j in range(len(s2)):
                if s1[i] != s2[j]:
                        return False
    return True
```

(II)
```
for i in range(len(s1)):
        if s1[i]==s2[i]:
                return True
     return False
```

(III) *for i in range(len(s1)):*
    if s1[i]!=s2[i]:
        return False
return True

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

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

def *makakilo(n):*
    ''' (int)->(int)
    Précondition: n>=1'''
    for i in range(n):
        total=0
        total=total + n
    return total

(a) Elle retourne n^n (b) Elle retourne factorial n (c) Elle retourne n*n
(d) Elle retourne n (e) Elle retourne 0

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

def *space(L,k):*
    if len(L)==0:
        return True
    return L[0]<k and space(L[1:],k)

(a) Retourne True si tous les éléments de L sont inférieurs 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 inférieur à k, et False sinon.
(d) Retourne True si le premier élément de L est inférieur à k, et False sinon.
(e) Exécute une boucle infinie.

29) Qu'est-ce que le programme Python suivant va afficher ?
def *maili(a, b):*
    if a % b == 0:
        return b
    else:
        return maili (b, a % b)

result= maili(44, 12)
print(result)

(a) 1 (b) 2 (c) 4 (d) 12 (e) 14

30) Quel est le nombre maximum d'appel de fonctions *maili* (précédente) s'exécutant à un moment donné au cours de l'exécution du programme précédent?
(a) 2 (b) 3 (c) 4 (d) 12 (e) 14

31) Combien de comparaisons effectuera une recherche binaire sur une liste triée de 1024 éléments si le nombre que nous recherchons n'est pas dans la liste ? Notez que tester un élément (pour voir s'il est plus grand que, égal à, ou inférieur à une valeur) est considérée comme l'une des comparaisons ici.
(a) au plus 6 (b) au plus 11 (c) au moins 512
(d) au moins 1024 (e) ne peut pas être determiné

32) 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 is_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 minuscule de l'alphabet Anglais

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

Lequel des algorithmes suivant(s) c.-à-d solution(s) peut être utilisé pour implémenter la function `is_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 (b) II (c) I et II (d) Aucune des déclarations ci-dessus.

33) Considérant le code suivant :

```
class Card:
    def __init__(self, rank, suit):
        '''(_____,str, str) -> None'''
        self.rank = rank
        self.suit = suit
```

Par quoi sera remplacé l'espace ____ dans le type des paramètres de la méthode *_init_()*?

(a) Deck (b) Card (c) str (d) None (e) Il n'est pas possible de dire.

34) Lequel des deux algorithmes suivant est plus efficace (dans le nombre d'opérations qu'il exige) pour résoudre le problème de trouver la somme du plus petit et du plus grand élément d'une liste L contenant au moins 100 élément. Les deux algorithmes sont corrects.

Algorithme A :
étape 1. Trier L utilisant un algorithme plus efficace, comme le tri par fusion (merge Sort)
étape 2. Retourner la somme du premier et du dernier nombre de la liste triée.

Algorithme B :
étape 1. Trouver la valeur minimale dans L de la manière la plus efficace
étape 2. Trouver la valeur maximale dans L de la manière la plus efficace
étape 3. Retourner la somme des deux valeurs trouvées à l'étape 1 et l'étape 2

(a) Algorithme A (b) Algorithme B (c) C'est impossible de dire

35) Supposons que vous avez une liste, a, contenant 2000 nombres qui sont classés du plus petit au plus grand et que vous voulez vérifier si un nombre particulier, x, apparaît dans a. Si vous faites ceci de la manière la plus efficace possible, vous pouvez le faire en effectuant des tests sur des valeurs de la liste et le nombre maximal de tests est égal à :

(a) au plus 1 (b) au plus 6 (c) 12 (d) au plus 1000 (e) au plus 2000

36) Supposons que vous avez une liste, a, contenant 2000 nombres qui qui ne sont pas triées dans un ordre particulier et que vous voulez trouver la valeur minimale dans a.
Si vous faites ceci de la manière la plus efficace possible, vous pouvez le faire en effectuant des tests sur des valeurs de la liste et le nombre maximal de tests est égal à :

(a) au plus 1 (b) au plus 6 (c) 12 (d) au plus 1000 (e) au plus 2000

37) Rappelons qu'un nombre premier est un entier positif supérieur à 1 qui n'a pas de diviseurs positifs autres que 1 et lui-même. La fonction suivante est elle correcte pour tout nombre entier n ≥2, comme décrit dans son docstring ?

```
def premier(n):
    '''(int)->bool
    Retourne True si n est premier, et False sinon.
    Précondition: n est un entier positif supérieur à 1'''

    est_premier =True
    i=2
    while i<n:
        if n%i == 0:
            est_premier =False
        i=i+1
    return est_premier
```

(a) Oui.
(b) Non, c'est faux quand n est 2.
(c) Non. Pour corriger, juste après la ligne *est_premier =`False`*, le fragment de code ci-dessous doit être ajouté de façon à ce que `else` soit alignée avec `if`.

```
else:
    est_premier =True
```

(d) Non. Pour corriger, la première ligne du corps de la fonction doit être remplacée par *est_premier =`False`* et la cinquième par *est_ premier =`True`*
(e) Non. Pour corriger, `while i<n:` devrait être remplacé par `while i<=n:`

38) Laquelle des fonctions suivantes calcule correctement et renvoie la somme 1/2+2/3+3/4+…+49/50?

```
def f1():
    sum = 0
    for i in range(1, 49):
        sum = sum+ i/(i + 1)
    return sum
def f2():
    sum=0
    for i in range(1, 50):
        sum = sum+ i/(i + 1)
    return sum
def f3():
    sum=0
    for i in range(49):
        sum = sum+(i+1)/(i + 2)
    return sum
```

(a) f2 et f3 (b) f1 et f3 (c) f1 (d) f2 (e) f3

39) La diagonale d'une matrice carrée commence dans le coin gauche supérieur jusqu'au coin droit inférieur. Par exemple, pour cette matrice carrée :
1 3 5
2 4 5
4 0 8
représentée par la liste imbriquée [[1, 3, 5], [2, 4, 5], [4, 0, 8]]. Les valeurs sur la diagonale sont 1, 4 et 8. Examiner la fonction suivante :

```
def get_diagonal_plus(L):
    '''(list of list of int) -> tuple of (list of int, bool)
    Retourne un tuple à deux éléments où le premier élément du tuple est une liste des
    valeurs sur la diagonale de la matrice carrée L. Le deuxième élément est True si
    toutes les valeurs en dehors de la diagonale sont nulles et False sinon.

    >>> get_diagonal_plus([[1, 3, 0], [2, 4, 5], [4, 0, 8]])
    ([1, 4, 8], False)
    >>> get_diagonal_plus([[24, 0, 0], [0, 0, 0], [0, 0, 0]])
    ([24, 0, 0], True)
    '''
    diagonal = []
    answer=True
    for row in range(len(L)):
        for col in range(len(L)):
            # CODE MANQUANT ICI
    return (diagonal, answer)
```

Lequel des fragments du code ci-dessous complète correctement cette fonction ?

(I)

```
if row == col:
    diagonal.append(L[row][col])
elif L[row][col]!=0:
    answer=False
```

(II)

```
if row == col:
    diagonal.append(L[row][col])
elif L[row][col]!=0:
    answer=False
else:
    answer=True
```

(III)

```
if row != col:
        if L[row][col]==0:
                answer=True
        else:
                answer=False
else:
        diagonal.append(L[row][col])
```

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

40) Considérer le code suivant:

```
def contains(value, lst):
        '''(object, 2D list) -> bool
        Retourne si la valeur est un élément d'une des listes imbriquées dans lst.
        >>> contains('moogah', [[70, 'blue'], [1.24, 90, 'moogah'], [80, 100]])
        True
        '''

        found = False # Nous n'avons pas encore trouvé de valeur dans la liste.
        # CODE MANQUANT ICI
        return found
```

Lequel des fragments du code ci-dessous complète correctement cette fonction ?

(I)

```
for sublist in lst:
        if value in sublist:
                found = True
```

(II)

```
for i in range(len(lst)):
        for j in range(len(lst[i])):
                if lst[i][j] == value:
                        found = True
```

(III)

```
for item in lst:
        if value == item:
                found = True
```

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

41) Rappelons que les conditions préalables (Préconditions) d'une fonction citent les conditions dans lesquelles cette fonction réalise correctement ce qui est énoncé dans la description (docstring) et ne va pas cracher ou s'exécuter infiniment.
Quelles sont les autres conditions préalables pour la fonction suivante ?

```
def pukalani(L):
''' # type manquant
Précondition: L ne contient que des entiers et
# le reste de ceci sont les conditions préalables manquantes
affiche tous les éléments de L jusqu'à ce qu'il rencontre un zéro ou un nombre négatif.
>>> pukalani([34,1,3,34,-7,4,55,-13,0, 100])
34 1 3 34
'''
    i=0
    while L[i]>0:
        print(L[i], end=' ')
        i=i+1
```

(a) aucune condition préalable n'est nécessaire, c.-à-d. la fonction fonctionne comme décrit pour toutes les listes L d'entiers.
(b) len(L) > 0
(c) len(L) > 1
(d) Les éléments de L ne sont pas tous positifs
(e) len(L) > 0 et les éléments de L ne sont pas tous positifs

42) Que réalise la fonction suivante ?

```
def coconut(L):
    '''(list of numbers)->bool
    Précondition: len(L)>=4'''

    L.sort()
    for i in range(len(L)-3):
        if L[i]==L[i+3]:
            return True
    return False
```

(a) Retourne True si le 1er et 3ème éléments dans L sont les mêmes, et False sinon.
(b) Retourne True si le 1er et 4ème éléments dans L sont les mêmes, et False sinon.
(c) Retourne True s'il y a au moins un élément qui apparait au moins 4 fois dans L et False sinon.
(d) Retourne True s'il y a au moins un élément qui apparait exactement 4 fois dans L et False sinon.
(e) Retourne True s'il y a au moins un élément qui apparait exactement 3 fois dans L et False sinon.

43) Considérez la classe suivante qui représente un point dans un plan.

```
class Point(object):
    def __init__(self, xcoord, ycoord):
        self.x = xcoord
        self.y = ycoord
        # CODE  MANQUANT
```

Imaginez qu'on veuille ajouter une méthode `distance` dans la classe `Point` (dans la partie # **CODE MANQUANT** qui retourne la distance du point jusqu'à l'origine du système de coordonnées (i.e., Point (0,0)). Exemple d'exécution:

```
>>> p=Point(0,1)
>>> p.distance()
1.0
```

Laquelle des solutions suivantes est correcte?

```
(I) def distance(self):
        return ((self.x ** 2) + (self.y ** 2)) ** 0.5
(II) def distance():
        return ((x ** 2) + (y ** 2)) ** 0.5
(III) def distance():
        return ((xcoord ** 2) + (ycoord ** 2)) ** 0.5
(IV) def distance(self):
        return ((self.xcoord ** 2) + (self.ycoord ** 2)) ** 0.5
```

(a) I (b) II (c) III (d) IV (e) aucune des options ci-dessus

44) Imaginez que les deux méthodes suivantes sont ajoutées à la classe *Point* précédante:

```
def __repr__(self):
        return "Point(" + str(self.x) + "," + str(self.y)+")"
def millieu(self, autre):
        mx = (self.x + autre.x) / 2
        my = (self.y + autre.y) / 2
        return Point(mx, my)
```

Si les lignes suivantes sont ajoutées comme programme principal, qu'est-ce que le programme suivant va afficher ?

```
p = Point(3, 4)
q = Point(5, 12)
m = p.millieu(q)
print(m)
```

(a) l'adresse de l'objet dans la mémoire. (b) Point(4.0,8.0) (c) (4.0,8.0)
(d) Point(3.5,8.5) (e) (3.5,8.5)

45) Considérez la défnition de la classe suivante qui utilise la classe `Point` de la question antérieure.

```
class Pointset(object):
    def __init__(self, points):
        '''(Pointset, list des objecs de type Point) -> None
        Initialise ce Pointset avec une liste de points.'''
        self.points = points
```

Lesquelles des lignes suivantes peuvent être utilisées pour créer un objet de classe `Pointset`?

(I) *points=[Point(1,2), Point(2,1), Point(0,0)]*
nouveau_pointset=Pointset(points)

(II) *nouveau_pointset=Pointset(Point(1,2), Point(2,1), Point(0,0))*

(III) *nouveau_pointset=Pointset([Point(0,0)])*

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

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

```
class Name:
    def __init__(self, firstName, mi, lastName):
        self.firstName = firstName
        self.mi = mi
        self.lastName = lastName

firstName = "John"
name = Name(firstName, 'F', "Smith")
firstName = "Peter"
name.lastName = "Pan"
print(name.firstName, name.lastName)
```

(a) Peter Pan (b) John Pan (c) Peter Smith (d) John Smith
(e) aucune des réponses ci-dessus

* Les questions suivantes (47-49) utilisent la classe suivante:

```
class Person:
        def __init__(self, first_name, last_name):
                self.first_name = first_name
                self.last_name = last_name
        def __eq__(self, other):
                return self.first_name == other.first_name and self.last_name ==
                other.last_name
      def __str__(self):
                return '(self.first_name, self.last_name)'
```

47) Qu'affiche le code suivant ?

```
pm1 = Person('Enna', 'Sicily')
pm2 = Person('Enna', 'Sicily')
print(pm1 == pm2, end=" ")
print(pm1 is pm2)
```

(a) True True (b) False True (c) True False (d) False False

48) Qu'affiche le code suivant ?

```
pm1 = Person('Enna', 'Sicily')
pm2 = pm1
print(pm1 == pm2, end=" ")
print(pm1 is pm2)
```

(a) True True (b) False True (c) True False (d) False False

49) Pour cette question on ajoute la classe suivante :

```
class Musician(Person):
        def __init__(self, first_name, last_name, instrument):
                super().__init__(first_name, last_name) #super() réfère à la clase de base
                self.instrument = instrument

pm1 = Person('Enna', 'Sicily')
pm2 = Musician('Enna', 'Sicily', 'trombone')
print(pm1 == pm2, end=" ")
print(pm1 is pm2)
```

Le cod va afficher :
(a) True True (b) False True (c) True False (d) False False

50) 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
   2 1
 3 2 1
4 3 2 1
5 4 3 2 1
6 5 4 3 2 1
```

(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):

51) Une partie de la fonction suivante manque.

```
def neg(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 (d) II (e) III

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

```
def gallerie(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 True sinon.
(e) Aucune des déclarations ci-dessus.

53) Pour la fonction `galleria` de la question précédente, combien d'opérations sont exécutées dans la fonction, approximativement, quand n devient large (n est le nombre d'éléments dans la liste L)?

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

54) Considérer la fonction suivante qui calcule x^n pour tout entier $n \geq 0$:

```
def puissance(x, n):
        '''Return x**n, fonctionne seulement pour les entiers non négatifs n'''

        if n <= 0: return 1
        return x * puissance(x, n-1)
```

Si nous appelons cette fonction avec n = 256, alors :
(a) cette fonction effectue 9 appels de fonctions récursives et 8 multiplications
(b) cette fonction effectue 9 appels de fonctions récursives et 255 multiplications
(c) cette fonction effectue 256 appels de fonctions récursives et 8 multiplications
(d) cette fonction effectue 256 appels de fonctions récursives et 256 multiplications
(e) cette fonction effectue 256 appels de fonctions récursives et 2^{255} multiplications

55) Considérer une autre version de cette fonction qui calcule x^n pour tout entier $n \geq 0$:

```
def puissance2(x, n):
   if n == 0: return 1
   res = puissance2(x, n//2) # x**(n//2)
   res = res*res # (x**(n//2)) * (x**(n//2)) = x**(2(n//2))
   if n % 2 == 1: #  n  est impair, et on veut avoir (x**(2(n//2)))*x = x**n
      res = res*x
   return res
```

Si nous appelons cette fonction avec n = 256, alors :

(a) cette fonction effectue 9 appels de fonctions récursives et 9 multiplications
(b) cette fonction effectue 9 appels de fonctions récursives et 255 multiplications
(c) cette fonction effectue 256 appels de fonctions récursives et 8 multiplications
(d) cette fonction effectue 256 appels de fonctions récursives et 255 multiplications
(e) cette fonction effectue 256 appels de fonctions récursives et 2^{255} multiplications

56) Considérer une autre version de cette fonction qui calcule x^n pour tout entier $n \geq 0$:

```
def puissance3(x, n):
    '''Retourne x**n, fonctionne seulement pour les entiers non négatifs de n'''
    if n == 0: return 1
    if n == 1: return x
    if n%2==0:
        return puissance3(x, n//2) * puissance3(x, n//2)
    else:
        return x* puissance3(x, n//2) * puissance3(x, n//2)
```

Si nous appelons cette fonction avec n = 128, alors :
(a) cette fonction effectue 9 appels de fonctions récursives
(b) cette fonction effectue 8 appels de fonctions récursives
(c) cette fonction effectue 256 appels de fonctions récursives
(d) cette fonction effectue 254 appels de fonctions récursives
(e) cette fonction effectue 255 appels de fonctions récursives