Université d'Ottawa
Faculté de génie

École de science informatique
et de génie électrique

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical Engineering
and Computer Science

ITI1520, Introduction à l'informatique, Examen final, Automne 2016

Durée: 3 hours
Date: le 15 décembre 2016, 14:00
Professeurs: Diana Inkpen, Aziz Abdesselam
Nombre de pages: 23

Instructions: À lire attentivement!

1. Ecrivez votre nom et votre numéro d'étudiant ici:

   ```
   Nom: _____________________________________________
   Numero d'etudiant _____________________________________
   ```

   et **complétez-les aussi sur la page Scantron avec des lettres/chiffres et en remplissant les cercles**.

2. Il faut soumettre ce papier avec les questions et la page Scantron avec les réponses.

3. Répondez à toutes les questions sur la page Scantron, de préférence avec un crayon pour pouvoir les modifier au besoin (stylo c'est bien aussi, mais pas de modifications). Remplissez bien les cercles pour les réponses, pour la correction automatique.

4. Choisissez seulement une réponse pour chaque question. Si vous pensez que plusieurs réponses sont correctes, choisissez la plus correcte/précise.

5. Tous les programmes sont en Python 3.

6. Cet examen est à livre fermé. Aucun livre ou appareil électronique (incluant les calculatrices) n'est permis.

7. Les points alloués à chaque question sont les mêmes.

8. Vous pouvez utiliser le verso des pages pour vos calculs et brouillons.

9. Regarder le questionnaire de l'un de vos voisins mènera à votre expulsion de la salle d'examen.

1. Quelle est la valeur de l'expression `4 % 7`?
   (a) 0 (b) 1 (c) 2 (d) 3 (e) 4

2. Lesquels des programmes suivants contiennent un appel de fonction?

   (I) `type(4.5)`

   (II)
   ```
   def ajoute_un(x):
       return x + 1
   ```

   (III) `aire(2, 9)`

   (IV) `print("Bonjour")`

   (a) seulement (III)
   (b) seulement (II) et (III)
   (c) seulement (I), (III), et (IV)
   (d) seulement (I), (II), (III), et (IV)
   (e) Aucune des déclarations ci-dessus.

3. Si `a` est un nombre entier de trois chiffres, lesquelles des lignes suivantes donnent le chiffre du milieu `a`? (Par exemple, si `a` est 456, le chiffre du milieu est 5.)

   (I) `(a // 10) % 10`

   (II) `(a % 100) // 10`

   (III) `(a % 10) // 10`

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

4. Qu'est-ce que le programme Python suivant va afficher sur l'écran?

```
vitesse=90
if vitesse > 120:
     print("Trop rapinde", end=' ')
elif vitesse < 120 or vitesse > 80:
     print("Bon intervalle!", end=' ')
elif vitesse == 90:
     print("Maintenir la vitesse.", end=' ')
if vitesse > 50:
     print("Vitesse correcte.", end=' ')
else:
     print("Trop lent.")
```

   (a) Maintenir la vitesse.
   (b) Bon intervalle! Maintenir la vitesse. Vitesse correcte.

(c) Maintenir la vitesse. Vitesse correcte.
(d) Bon intervalle!
(e) Bon intervalle! Vitesse correcte.

5. Voici un tableau des salaires horaires.

	experience: 0 année	experience: 1 ou 2 années	experience: 3 ou plus
age: moins que 18:	$6.50	$9.5	$11.00
age: 18 ou plus	$6.50	$12.00	$12.00

Dans le programme ci-dessus, (1) et (2) doivent être remplacés par des expressions booléennes. Quelles sont les expressions correctes? Les variables `experience` et `age` ont des valeurs entieres.

```
if experience == 0:
   salaire = 6.5
elif (1) :
   if (2) :
      salaire = 9.5
   else:
      salaire = 11
else:
   salaire = 12
```

(a) (1) `experience > 0` , (2) `experience >= 1`
(b) (1) `age < 18` , (2) `experience != 3`
(c) (1) `age < 18` , (2) `experience == 1 or experience == 2`
(d) Aucune des expressions ci-dessus.

6. La fonction Python `sort` prend une liste et ne retourne rien. Elle modifie la liste pour qu'elle soit triée en ordre croissant (du plus petit au plus grand). Son docstring est: (list) -> None
Qu'est-ce que le programme Python suivant va afficher sur l'écran?

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

(a) 18
(b) 34
(c) None
(d) Rien. Le programne donne une erreur.

7. On vient de définir une classe `Chose` avec une méthode appelée `foo`. Le corps de la méthode `foo` et les autres méthodes ne sont pas donnés pour économiser de l'espace.

```
class Chose:

   def foo(self, a):
      ...
```

Si `t` est une variable (un objet) de classe `Chose` et `d` est une autre variable (initialisée avec n'importe quelle valeur), comment faut-il appeler la méthode `foo`?

(a) `foo(d)`
(b) `t.foo(d)`
(c) `foo(self, d)`
(d) `foo(t, d)`
(e) `t.foo(t, d)`

8. A, B et C sont des variables booléennes. On veut écrire une expression booléenne qui prend la valeur True si une ou plusieures des trois variables sont fausses (ont la valeur False). Dans le cas contraire, l'expression devrait prendre la valeur False. Lesquelles des expressions suivantes sont correctes?

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

9. Qu'est-ce que le programme Python suivant va afficher sur l'écran?

```
def foo(x):
     i=0
     while i < len(x):
          print(x[i], end=" ")
          while (i<len(x)-1 and x[i] == x[i+1] ):
               i=i+1
          i=i+1
#programme principal
foo([3,-4,-4,5,5,5,7,0,0,0,0,6])
```

(a) `3 -4 5 5 7 0 0 6`
(b) `3 -4 5 5 7 0 0`
(c) `3 -4 5 7 0 6`
(d) `3 -4 5 7 0`
(e) Le programme affiche quelque chose et donne un message d'erreur (index out of range).

10. Dans la question précédente, combien d'arguments (paramètres actuels) y a-t-il dans l'appel `foo`?

(a) `0`
(b) `1`
(c) `2`
(d) `12`
(e) Aucune des valeurs ci-dessus.

11. Que fait la fonction Python suivante?

```
def ma_fonction(a):
     '''(list of int) -> None
     Precondition: len(a) >= 3'''

     while len(a) > 3:
          a.remove(min(a))
```

Rappelez-vous que la fonction Python `min` retourrne le minimum dans une liste, et la fonction Python `remove` élimine la première occurrence d'une valeur dans la liste (elle modifie la liste). Voici le `help` (docstring) de Python pour cette fonction:

```
remove(...)
    L.remove(value) -> None -- remove first occurrence of value.
```

(a) Modifie la liste pour éliminer les `len(a)-4` plus petites valeurs de la liste `a`.
(b) Modifie la liste `a` pour retenir seulement les 3 valeurs plus grandes.
(c) Modifie la liste `a` pour retenir seulement les 3 valeurs plus petites.
(d) Modifie la liste `a` pour retenir seulement les 4 valeurs plus grandes.
(e) Aucune des déclarations ci-dessus.

12. 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 élements 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 élements de `L` est inférieur à `k`, et False sinon.
(d) Retourne True si le premier élement de `L` est inférieur à `k`, et False sinon.
(e) Execute une boucle infinie.

13. Lesquels des programmes suivants I, II, et III affichent le même message que ce programme?

```
x = 10
if x < 30:
    print(x, "est inferieur a 30")
else:
    if x > 30:
        print(x, "est superieur a 30")
    else:
        print(x, "est 30")
```

I

```
x=10
if x < 30:
    print(x, "est inferieur a 30")
else x > 30:
    print(x, "est superieur a 30")
else:
    print(x, "est 30")
```

II

```
x=10
if x < 30:
    print(x, "est inferieur a 30")
elif x > 30:
    print(x, "est superieur a 30")
else:
    print(x, "est 30")
```

III

```
x=10
if x < 30:
    print(x, "est inferieur a 30")
if x > 30:
    print(x, "est superieur a 30")
else:
    print(x, "est 30")
```

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

14. Lesquelles des déclarations suivantes sont correctes au sujet de la fonction orange (qui prend comme paramètre un entier positif)?

```
def orange(num):
    ''' (int)->int
    Precondition: num est positif '''

    t = 0
    for i in range(num):
        t = t + 10
    return t
```

(a) La fonction calcule la valeur de 10 plus `num`.

(b) La fonction calcule the valeur 10 à l'exposant `num`.

(c) La fonction calcule 10 multiplié par `num`.

(d) La fonction calcule le factoriel de `num`

(e) Aucune des déclarations ci-dessus

15. Quelle est la description correcte de la fonction kiwi?

```
def kiwi(x):
     '''(list)->bool
        Precondition: les elements de la liste x sont des nombres
        et x a au moins 2 elements
     '''
     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 retourne True si les nombres de la liste x sont en ordre croissant, du plus petit au plus grand, et False sinon.

(b) La fonction retourne True si les nombres de la liste x sont en ordre décroissant, du plus grand au plus petit, et False sinon.

(c) La fonction retourne False si l'avant-dernier élément est supérieur au dernier élément, et True sinon.

(d) Aucune des déclarations ci-dessus.

16.
```
if heure >=0 and heure <=12:
    return False
 else:
    return True
```

Lesquelles des déclarations suivantes sont équivalentes au code ci-dessus?

I) `return heure >=0 and heure <=12`
II) `return not(heure >=0 and heure<=12)`
III) `return heure <0 or heure >12`

(a) I seulement

(b) II seulement

(c) III seulement

(d) I et III

(e) II et III

17. Voici une fonction qui trie une liste `a` (du plus petit au plus grand):

```
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]
```

Si la liste `a` contient 10 000 élements, la ligne 3 de `bubble_sort(a)` s'éxecute

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

18. Si la liste `a` contient 10 000 élements *qui sont déjà triés du plus petit au plus grand*, la ligne 4 de `bubble_sort(a)` s'éxecute

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

19. Considerez la fonction suivante:

```
def ma_fonction(n):
     '''(int)->None
     Precondition: n est un entier positif
     '''
     for i in range(1,n+1):
          if(n%i == 0):
               print(i, end=" ")
     print()
```

Que fait la fonction?

(a) affiche tous les entiers premiers inférieurs ou égaux à `n`
(b) affiche tous les entiers non premiers inférieurs ou égaux à `n`
(c) affiche tous les entiers divisibles par `i`
(d) affiche tous les diviseurs de `n+1`
(e) affiche tous les diviseurs de `n`

20. Qu'est-ce que le programme Python suivant va afficher sur l'écran?

```
desserts = [['gateau', 'fromage'], ['creme', 'brulee'], ['chocolat']]
print(desserts[1][1])
```

(a) gateau
(b) fromage
(c) creme
(d) brulee
(e) chocolat

21. Qu'est-ce que le programme Python suivant va afficher sur l'écran?

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

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

(a) Affiche 250 seulement.
(b) Affiche 10 seulement.
(c) Affiche 100 seulement.
(d) Affiche 10 et 100.
(e) Affiche 10 et donne une erreur.

22. Qu'est-ce que le programme Python suivant va afficher sur l'écran?

```
def sara(x, y):
     print(x+y)
     return x+y

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

(a) Affiche 250 seulement.
(b) Affiche 10 seulement.
(c) Affiche 100 seulement.
(d) Affiche 10 et 100.
(e) Affiche 10 et donne une erreur.

23.
```
a = [1, 2, 3, 4]
b = a
a[1] = 0
print(b)
```

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

24.
```
a = [1, 2, 3, 4]
b = a[:]  #rappelez-vous que a[:] produit une copie de la liste a
a[1] = 0
print(b)
```

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

25. Qu'est-ce que le programme Python suivant va afficher sur l'écran?

```
class Point(object):
    def __init__(self, xcoord=0, ycoord=0):
        self.x = xcoord
        self.y = ycoord
```

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

def f(x,a,b):
    a=b
    a.x=1000
    a.y=1000
    x=x+3

x=0
p1=Point(1,2)
p2=Point(10,20)
f(x,p1,p2)
print(x, p1, p2)
```

(a) va afficher `0 Point(1000,1000) Point(1000,1000)`

(b) va afficher `3 Point(1000,1000) Point(1000,1000)`

(c) va afficher `3 Point(1,2) Point(1000,1000)`

(d) va afficher `0 Point(1000,1000) Point(10,20)`

(e) va afficher `0 Point(1,2) Point(1000,1000)`

26. Le corps de la fonction suivante manque.

```
def meme_chaine(s1,s2):
     '''(str,str)->bool
     Precondition: len(s1)==len(s2) > 1
     Pour deux chaines de caracteres s1 et s2 de meme taille,
     la fonction retourne True si les deux chaines
     contiennent les memes caracteres dans la meme orde et False sinon.
     '''
```

Quel code doit-on y ajouter pour que la description soit correcte?

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

27. Une partie de la fonction suivante manque.

```
def common_chars(s1, s2):
    '''(str, str) -> str

    Retourne une nouvelle chaine avec les caracteres de s1 qui arrivent
    au moins une fois en s2. Les caracteres dans le resultat conservent
    leur ordre d'occurrence en s1.

    >>> common_chars('abc', 'ad')
    'a'
    >>> common_chars('a', 'a')
    'a'
    >>> common_chars('abb', 'ab')
    'abb'
    >>> common_chars('abracadabra', 'ra')
    'araaara'
    '''

    res = ''

    # code manque

    return res
```

Quel est le fragment de code qui manque?

(a)
```
for ch in s1:
    for ch in s2:
        res = res + ch
```

(b)
```
for ch in s2:
    if ch in s1:
        res = res + ch
```

(c)
```
for ch in s1:
    if ch in s2:
        res = res + ch
```

(d)
```
if ch in s2:
        for ch in s1:
            res = res + ch
```

28. Quelle est la description correcte de la fonction suivante?

```
def ma_fonction(n):
     ''' (int)->(int)
     Precondition: n>=1'''
     for i in range(n):
          total=1
          total=total*n
     return total
```

(a) elle retourne n^n

(b) elle retourne $n * n$

(c) elle retourne n

(d) elle retourne 1

(e) elle donne erreur pour $n \geq 1$

29. Qu'est-ce que le programme Python suivant va afficher sur l'écran?

```
def ma_fonction(a, b):
    if a % b == 0:
         return b
    else:
      return ma_fonction(b, a % b)

resultat = ma_fonction(44, 12)
print(resultat)
```

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

30. Pour le programme de la question précédente, combien d'appels à ma_fonction sont effectués? Autrement dit, quel est le nombre maximal des fonctions qui sont executées en même temps?

(a) 2 (b) 3 (c) 4 (d) 12 (e) 14

31. Considérez le programme suivant:

```
def xfonction(x, n):
  if n > 0:
    print(x[n - 1], end = " ")
    xfonction(x, n-1)

x = [1, 2, 3, 4, 5]
xfonction(x, 5)
```

Laquelle des déclarations suivantes est correcte?

(a) Le programme affiche 1 2 3 4 5.

(b) Le programme affiche 1 2 3 4 5 et donne un message d'erreur (index out of range).

(c) Le programme affiche 5 4 3 2 1.

(d) Le programme affiche 5 4 3 2 1 et donne un message d'erreur (index out of range).

(e) Aucune des déclarations ci-dessus.

32. Une chaine de caractères s1 est un anagramme de s2 si les lettres de s1 peuvent être réarrangées pour donner s2. Par exemple, "aride" est un anagramme de "radie". Considérez la fonction suivante:

```
def est_anagramme(s1, s2):
    ''' (str, str) -> bool
    Retourne True si s1 est un anagramme de s2, et False sinon.
    Precondition: les caracteres en s1 et s2 sont des lettres en minuscules

    >>> est_anagramme("ab", "a")
    False
    >>> est_anagramme("aimer", "marie")
    True
    >>> est_anagramme("bonjour", "bonjour")
    True
    '''
```

Lesquels des algorithmes suivants peuvent être utilisés pour implémenter la fonction `est_anagramme`? (Tracez-les pour vous-même pour les exemples ci-dessus.)

(I) Pas 1. Transformez s1 dans une liste de caractères L1. (e.g., si `s1='abba'`, `L1=['a','b','b','a']`
Pas 2. Transformez s2 dans 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 dans une liste de caracteres L1.
Pas 2. Transformez s2 dans une liste de caracteres L2.
Pas 3. Triez les deux listes. (trier une liste de caractères va arranger les caracteres 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 déclarations ci-dessus.

33. Quelle est la description correcte de la fonction suivante?

```
def pomme(L, c):
   """ (list, str) -> int
```

```
    Precondition: L est une liste de chaines de caracteres.
    Chaque element de L contient au moins un caractere.
    len(c) == 1, i.e., c contient un seul caractere
    """

    res = []

    for elem in L:
        if elem[0] == c:
            res.append(elem)

    return len(res)
```

(a) Retourne la liste de chaines de caracteres qui commence avec c.

(b) Retourne le nombre total de caracteres de toutes les chaines de L.

(c) Retourne le nombre de chaines de caracteres de L qui contiennent c.

(d) Retourne le nombre de chaines de caracteres qui commencent avec c.

(e) Aucune des déclarations ci-dessus.

34. Considérez le programme suivant:

```
class Carte(object):
    def __init__(self, couleur, valeur):
        ''' (______,str, str) -> None'''
        self.couleur = couleur
        self.valeur = valeur
```

Avec quoi remplacer la partie `______` dans le constructeur?

(a) Point

(b) Carte

(c) str

(d) None

(e) L'information disponible n'est pas suffisante pour décider.

35. Lequel des deux algorithmes suivants est le plus efficace (en nombre d'opérations nécessaires) pour trouver la somme du plus petit et du plus grand nombre de la liste? La liste L contient au moins 1000 nombres. Chacun des deux algorithmes est une solution correcte. Mais lequel est le plus efficace?

Algorithme A:
Pas 1. triez la liste L avec tri par fusionnement (merge sort)
Pas 2. retournez la somme du premier et du dernier nombre de la liste triée.

Algorithme B:
Pas 1. trouvez le nombre minimum en L
Pas 2. trouvez le nombre maximum en L
Pas 3. retourne la somme des deux nombres trouvés dans Pas 1 et Pas 2

(a) Algorithme A
(b) Algorithme B
(c) L'information disponible n'est pas suffisante pour décider.

36. Imaginez qu'on a une liste `a` qui contient 1000 éléments en ordre croissant (du plus petit au plus grand) et on veut vérifier si une valeur `x` existe dans `a`. Si on choisit l'algorithme le plus efficace, combien va-t-on inspecter d'éléments de `a`? (Rappelez-vous que la recherche binaire est la plus efficace méthode de recherche, si la liste est triée. On cherche la valeur au millieu de la liste. Si la valeur n'est pas là, on cherche dans la moitié gauche de la liste, si la valeur est inferieure à la valeur au millieu; sinon dans la moitié doite. Le processus continue jusqu'à ce que la moitié où on cherche soit vide.)
(a) au plus 1
(b) au plus 5
(c) au plus 10
(d) au moins 100
(e) au moins 1000

37. Imaginez qu'on a une liste `a` qui contient 1000 éléments qui ne sont pas arrangés (triés) et on veut vérifier si une valeur `x` existe dans `a`. Si on choisit l'algorithme le plus efficace, combien va-t-on inspecter d'éléments de `a`?
(a) au plus 1
(b) au plus 5
(c) au plus 10
(d) au moins 100
(e) au moins 1000

38. Lesquelles des fonctions suivantes calculent correctement et retournent la somme $1/2 + 2/3 + 3/4 + ... + 99/100$?

```
def f1():
    somme = 0
    for i in range(1, 99):
        somme = somme + i / (i + 1)
    return somme

def f2():
    somme = 0
    for i in range(1, 100):
        somme = somme + i / (i + 1)
    return somme

def f3():
    somme = 0
    for i in range(99):
        somme = somme + (i+1) / (i + 2)
    return somme
```

(a) f1 seulement
(b) f2 seulement
(c) f3 seulement
(d) f1 et f3
(e) f2 et 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 2D `[[1, 3, 5], [2, 4, 5], [4, 0, 8]]`, les valeurs de la diagonale sont 1, 4 et 8.

Considérez la fonction suivante, avec une partie de code qui manque:

```
def calcule_diagonale_et_non_diagonale(L):
    '''(list) -> tuple de (list, list)
    Precondition: Les listes sont des listes 2D des entiers.
    Retourne un tuple avec 2 elements. Le premier element est une liste avec des
    valeurs de la diagonale de la matrice carree L et le deuxieme element est
    une liste avec les autres valeurs de L.

    >>> calcule_diagonale_et_non_diagonale([[1,  3,  5], [2,  4,  5], [4,  0,  8]])
    ([1, 4, 8], [3, 5, 2, 5, 4, 0])
    '''
    diagonale = []
    non_diagonale = []
    for r in range(len(L)):
       for c in range(len(L)):

         # CODE MANQUE

     return (diagonale, non_diagonale)
```

Lesquels des fragments de code suivants sont corrects pour compléter cette fonction?

I

```
if r == c:
    diagonale.append(L[r][c])
else:
    non_diagonale.append(L[r][c])
```

II

```
if r == c:
    diagonale.append(L[r][c])

non_diagonale.append(L[r][c])
```

III

```
if r == c:
    diagonale.append(L[r][c])
if r != c:
    non_diagonale.append(L[r][c])
```

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

40. Considérez le programme suivant:

```
def contient(val, lst):
    '''(object, list) -> bool
    Retourne True si val est un element d'une des listes de la liste 2D lst.

    >>> contient('moogah', [[70, 'bleu'], [1.24, 90, 'moogah'], [80, 100]])
    True
    '''
    trouve = False  #  On n'a pas encore trouve val dans la liste.

    # CODE MANQUE

    return trouve
```

Lesquels des fragments de code suivants sont corrects pour compléter cette fonction?

I

```
for sousListe in lst:
    if val in sousListe:
        trouve = True
```

II

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

III

```
for i in lst:
    if i == val:
        trouve = True
```

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

41. Quelle est la description correcte de la fonction suivante?

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

(a) Retourne True si le premier et le troisième éléments de L sont égaux, et False sinon.

(b) Retourne True si le premier et le quatrième éléments de L sont égaux, et False sinon.

(c) Retourne True s'il y a au moins 4 éléments de L qui sont égaux, et False sinon.

(d) Retourne True s'il y a exactement 4 éléments de L qui sont égaux, et False sinon.

42. 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 manque
```

Imaginez qu'on veuille ajouter une méthode `distance` dans la classe Point (dans la partie `#code manque`) 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

43. Imaginez que les deux méthodes suivantes sont ajoutées à la classe Point de la question precedante:

```
    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 suivante sont ajoutées comme programme principal:

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

Qu'est-ce que le programme va afficher?

(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)`

44. Considérez la définition 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 le Pointset avec la liste 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 seulement
(b) II seulement
(c) III seulement
(d) I et II
(e) I et III

45. Qu'est-ce que le programme Python suivant va afficher sur l'écran?

```
class Nom(object):
    def __init__(self, prenom, surnom):
        self.surnom = surnom
        self.prenom = prenom

prenom = "Jean"
nom1 = Nom(prenom, "Valjean")
prenom = "Peter"
nom1.surnom = "Pan"
print(nom1.prenom, nom1.surnom)
```

(a) Jean Valjean
(b) Jean Pan
(c) Peter Pan
(d) Peter Valjean
(e) aucune des options ci-dessus

* Les questions suivantes utilisent la classe suivante:

```
class Personne(object):
    def __init__(self, prenom, surnom):
        self.prenom = prenom
        self.surnom = surnom
    def __eq__(self, autre):
        return self.prenom == autre.prenom \
               and self.surnom == autre.surnom
    def __str__(self):
        return '(self.surnom, self.prenom)'
```

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

```
pm1 = Personne('Eve', 'LeBrun')
pm2 = Personne('Eve', 'LeBrun')
print(pm1 == pm2, end=" ")
print(pm1 is pm2)
```

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

47. Qu'est-ce que le programme suivant va afficher?

```
pm1 = Personne('Eve', 'LeBrun')
pm2 = pm1
print(pm1 == pm2, end=" ")
print(pm1 is pm2)
```

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

48. Pour cette question, on ajoute la classe suivante:

```
class Musicien(Personne):
    def __init__(self, prenom, surnom, instrument):
        super().__init__(prenom, surnom)
        self.instrument = instrument
```

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

```
pm1 = Personne('Eve', 'LeBrun')
pm2 = Musicien('Eve', 'LeBrun', 'trombone')
print(pm1 == pm2, end=" ")
print(pm1 is pm2)
```

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

49. 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):`

50. 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 manque

     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

51. 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 trie 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 element
               return True
          elif i==len(L)-1 and X[i]!=X[i-1]: # X[i] est le dernier element
               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 elements de `L` sont tous differents (i.e., si `L` ne contient pas de doubles), et False sinon.

(b) Retourne False si les elements de `L` sont tous differents (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 déclarations ci-dessus.