+1/1/60+

EPFL

1

Ens. : O. Lévêque, M. Stojilović
Information, Calcul, Communication - A
Vendredi 19 décembre 2025
Durée : 180 minutes

U.N.Owen

SCIPER : **0**

Salle : INF 1

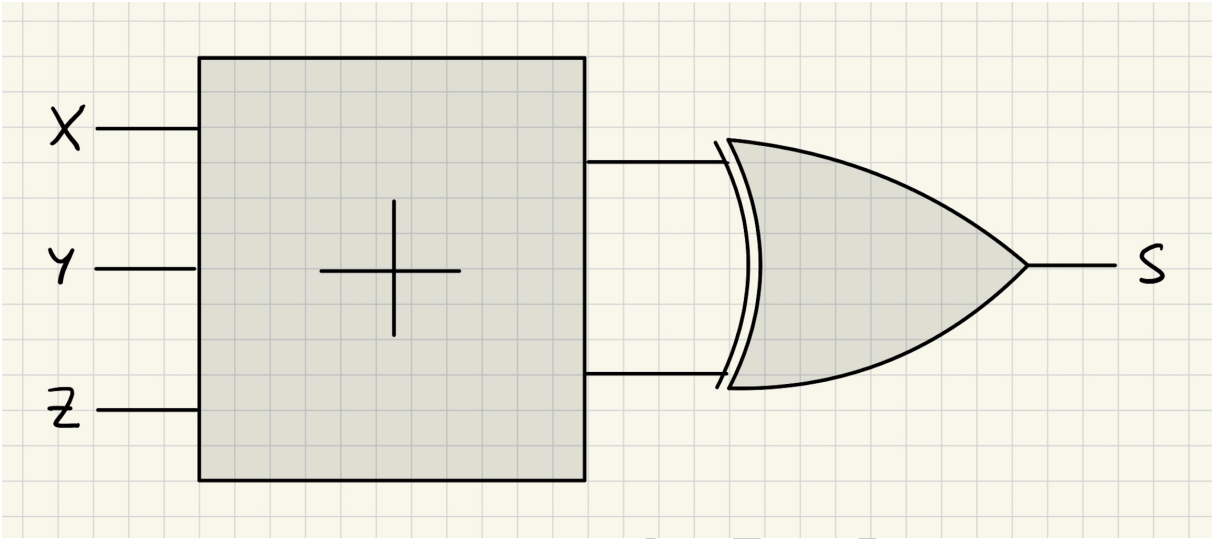
Attendez le début de l'épreuve avant de tourner la page. Ce document est imprimé recto-verso, il contient 20 pages, les dernières pouvant être vides. Ne pas dégrafer.

- Posez votre carte d'étudiant sur la table.
- Document autorisé pour cet examen : un formulaire constitué de deux pages A4 recto-verso, manuscrites, préparées avec stylet+tablette ou à l'ordinateur.
- Seule l'utilisation d'une calculatrice simple (de type TI-30 eco RS) est autorisée pendant l'examen. Tout autre appareil électronique (ordinateur, smartphone/watch, tablette) est interdit.
- L'examen est composé uniquement de questions de type ouvert, valant en tout 40 points.
- Merci d'avance de soigner la présentation de vos réponses !
- Laissez libres les cases à cocher : elles sont réservées aux correcteurs.
- Si une question est erronée, les enseignants se réservent le droit de l'annuler.

Question 1: *Cette question est notée sur 5 points.*

$\square_0$ $\square_1$ $\square_2$ $\square_3$ $\square_4$ $\square_5$

On considère le circuit suivant :



où le premier circuit à gauche est l'additionneur de 3 bits utilisé comme "brique de base" dans l'additionneur de nombres entiers vu au cours.

a) (2 points) Pour quelles valeurs de X, Y, Z en entrée la sortie S du circuit ci-dessus vaut-elle 0 ?

b) (3 points) Proposez un circuit alternatif composé uniquement de portes ET, OU et NON, dont la sortie S soit identique à celle du circuit de la page ci-contre (pour les mêmes entrées X, Y, Z).

Votre circuit ne doit pas comporter plus de 7 portes au total.

Question 2: *Cette question est notée sur 5 points.*

$\square_0$ $\square_1$ $\square_2$ $\square_3$ $\square_4$ $\square_5$

Soit $n \geq 1$ un nombre entier positif. On considère le signal suivant :

$$X(t) = \sum_{k=1}^{n} \sin(2^k \pi t)$$

où le temps t est mesuré en secondes.

a) (1 point) Que vaut la bande passante du signal X ? (exprimée en Hertz)

Le signal X est filtré avec un filtre à moyenne mobile de période d'intégration $T_c = 2^{-k_0}$ seconde, où k_0 est un nombre entier compris entre 1 et n (inclus).

b) (2 points) Que vaut le signal $\widehat{X}$ à la sortie du filtre et combien de composantes de ce signal ont-elles une amplitude non-nulle ?

c) (2 points) Supposons maintenant qu'on échantillonne le signal filtré $\widehat{X}$ à une fréquence f_e comprise strictement entre 2^{k_0} et 2^{k_0+1} Hertz et qu'on reconstruise ensuite celui-ci grâce à la formule d'interpolation vue en cours.

Le signal reconstruit sera-t-il identique au signal d'origine X, au signal filtré $\widehat{X}$, ou différent de ces deux signaux ? Justifiez pleinement votre réponse.

Question 3: *Cette question est notée sur 5 points.*

☐0 ☐1 ☐2 ☐3 ☐4 ☐5

a) (2 points) Proposez un encodage binaire pour la séquence de 16 chiffres

2 4 6 8 1 0 1 2 1 4 1 6 1 8 2 0

qui respecte les trois conditions suivantes :

- à chaque chiffre correspond un unique mot de code ;
- la séquence binaire ainsi produite doit être décodable de manière univoque ;
- le nombre total de bits utilisés doit être minimal ;

et dessinez également l'arbre binaire que vous utilisez pour l'encodage.

b) (2 points) Exprimez l'entropie de la séquence de chiffres sous la forme $H = a + b\,\log_2(3) + c\,\log_2(5)$, où a, b, c sont des nombres rationnels, puis calculez sa valeur numérique. Votre résultat est-il cohérent avec la partie a) ?

c) (1 point) On propose maintenant d'encoder la séquence avec le dictionnaire suivant :

2 : 10, 4 : 100, 6 : 110, 8 : 1000, 10 : 1010, 12 : 1100, 14 : 1110, 16 : 10000, 18 : 10010, 20 : 10100

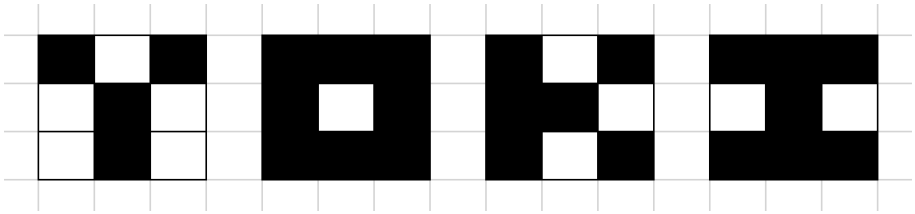
(où on encode donc certains chiffres par groupes de deux plutôt qu'isolément).

Ce système d'encodage est-il meilleur ou moins bon que celui que vous avez proposé plus haut ? Justifiez votre réponse.

Question 4: *Cette question est notée sur 5 points.*

☐0 ☐1 ☐2 ☐3 ☐4 ☐5

Pour transmettre le message “YOKI” composé de 4 lettres, on utilise pour chaque lettre une grille de 3×3 pixels noirs et blancs :

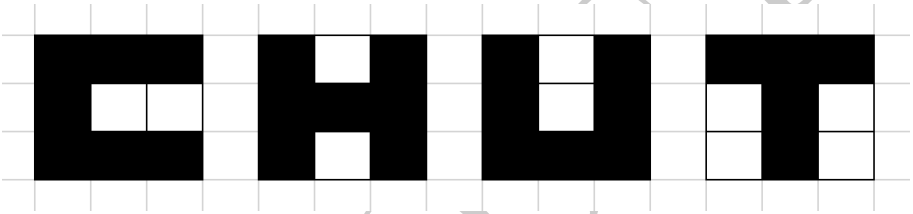


et on suppose qu’on n’utilise que ces 4 lettres pour la transmission.

a) (2 points) Supposons que lors de la transmission, certains pixels soient effacés (on peut imaginer ces pixels devenant gris, par exemple). Jusqu’à combien d’effacements par lettre, dans le pire des cas, le décodage du message est-il toujours possible sans ambiguïté ? Justifiez votre réponse.

b) (1 point) Si maintenant ce ne sont pas des effacements qui surviennent lors de la transmission, mais des erreurs du type : un pixel noir devient blanc, ou l'inverse. Jusqu'à combien d'erreurs par lettre, dans le pire des cas, le décodage du message est-il toujours possible sans ambiguïté ? A nouveau, justifiez votre réponse.

c) (2 points) Finalement, si au lieu du message "YOKI", c'est le message "CHUT" qui est transmis :



et on suppose à nouveau qu'on n'utilise que ces 4 lettres pour la transmission.

Comment les réponses aux questions a) et b) seront-elles modifiées dans ce cas ?

Question 5: *Cette question est notée sur 1.5 points.*

☐0 ☐.5 ☐1 ☐.5

Qu'affiche ce programme ?

```
a = set("letter")
b = set("coffee")

def combine_ver1(x):
    b = set("butter")
    print(sorted(x | b))

def combine_ver2(x):
    global b
    b = set("banana")
    print(sorted(b ^ x))

combine_ver2(a)
combine_ver1(b)
print(sorted(b))
```

Fournissez uniquement la réponse demandée ; aucune étape intermédiaire n'est nécessaire. La totalité des points ne sera attribuée que si votre réponse correspond exactement à ce qu'affiche le programme dans le terminal. Donner uniquement la valeur calculée ne suffit pas.

Question 6: *Cette question est notée sur 1.5 points.*

☐0 ☐.5 ☐1 ☐.5

Qu'affiche ce programme ?

```
def bar(arr):
    a = len(arr)
    b = len(arr[0])

    result = {}
    for c in range(b):
        s = 0
        for r in range(a):
            s += arr[r][c]
        result[c + 1] = s
        if s < 0:
            break

    return result

m = [[ 0,  1, -1,  0],[ 2,  2,  0, -1],[-1, -1, -2,  3]]
print(bar(m))
```

Fournissez uniquement la réponse demandée ; aucune étape intermédiaire n'est nécessaire. La totalité des points ne sera attribuée que si votre réponse correspond exactement à ce qu'affiche le programme dans le terminal. Donner uniquement la valeur calculée ne suffit pas.

Question 7: *Cette question est notée sur 1.5 points.*

☐0 ☐.5 ☐1 ☐.5

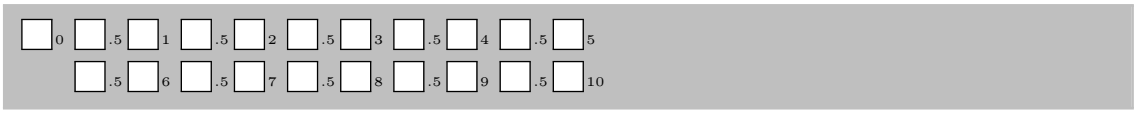
Qu'affiche ce programme ?

```
def num_L(n):
    if n <= 0:
        return 2
    if n == 1:
        return 1
    return num_L(n - 1) + num_L(n - 2)

print(num_L(5))
print(num_L(6))
```

Fournissez uniquement la réponse demandée ; aucune étape intermédiaire n'est nécessaire. La totalité des points ne sera attribuée que si votre réponse correspond exactement à ce qu'affiche le programme dans le terminal. Donner uniquement la valeur calculée ne suffit pas.

Question 8: *Cette question est notée sur 10 points.*



De nombreux numéros d'identification (par exemple les numéros de carte de crédit) utilisent une *checksum* pour détecter des erreurs de frappe. L'*algorithme de Luhn* est une méthode simple pour calculer une telle checksum. Dans cet exercice, nous utilisons l'algorithme de Luhn sur une chaîne de chiffres `s`, comme suit :

1. On part du chiffre le plus à droite de `s` et on se déplace vers la gauche.
2. On numérote les positions à partir de 0 :
   - chiffre le plus à droite : position 0,
   - chiffre suivant vers la gauche : position 1, etc.
3. Pour chaque chiffre :
   - si sa position est paire, on conserve le chiffre tel quel ;
   - si sa position est impaire, on double le chiffre ; si le résultat est ≥ 10, on lui soustrait 9.
4. On additionne tous ces nombres. On note cette somme `total`.
5. La checksum est simplement le reste de la division entière de `total` par 10.

Vous pouvez supposer que toutes les chaînes ne contiennent que des caractères chiffres (`"0"`–`"9"`).

Exemple : Soit `s = "1057"`. On numérote les positions en partant de la droite : `"7"` est en position 0 (paire, on garde le chiffre 7), `"5"` est en position 1 (impaire, on le double à 10 puis, comme le résultat est ≥ 10, on lui soustrait 9 et on obtient 1), `"0"` est en position 2 (paire, on garde le chiffre 0), et `"1"` est en position 3 (impaire, on le double à 2). La somme vaut donc $7+1+0+2=10$, puis la checksum vaut $10 \mod 10 = 0$; lorsque la checksum est égale à zéro, le numéro est considéré comme *valide*. À titre de comparaison, si l'on numérote les positions en partant de la gauche, on obtient une somme 11 et une checksum $11 \mod 10 = 1$, ce qui indiquerait (à tort) que le numéro n'est pas valide.

Considérez le dictionnaire suivant `id_numbers`, qui contient les numéros d'identification pour lesquels nous voulons calculer la checksum.

```
1  id_numbers = {
2      "Amelie":    "12344",
3      "Benoit":    "3456780",
4      "Chloe":     "79920",
5      "Dominique": "567891",
6      "Elodie":    "810239",
7  }
```

Pour chaque sous-question où l'on vous demande d'écrire une fonction, vous devez fournir la définition complète de la fonction, et non seulement le corps. Le non-respect des consignes spécifiques d'implémentation indiquées ci-dessous (par exemple l'utilisation des fonctions imposées ou des constructions demandées) entraînera l'attribution d'au plus 50% des points prévus pour la question concernée, voire moins selon le nombre et la nature des écarts par rapport aux consignes.

(a) `string_to_digits`

Écrivez une fonction `string_to_digits` qui prend en argument une chaîne `number` et qui renvoie une liste d'entiers correspondant aux chiffres de `number`.

Vous devez construire cette liste à l'aide d'une compréhension de liste. L'ordre des chiffres dans votre liste doit être le même que dans la chaîne d'origine. Vous ne devez pas utiliser d'indexation du type `number[i]` et vous ne devez pas appeler `list(number)`.

Exemple :

```
1  string_to_digits("12344")   # returns [1, 2, 3, 4, 4]
```

(b) `luhn_checksum`

(b1.) Écrivez une fonction `luhn_checksum` qui prend en argument une chaîne `number`, appelle la fonction `string_to_digits` pour obtenir la liste des chiffres, calcule la Luhn checksum sur cette liste de chiffres, et renvoie la checksum sous la forme d'un entier entre 0 et 9.

Vous devez utiliser la fonction `string_to_digits` à l'intérieur de `luhn_checksum` pour convertir la chaîne en liste de chiffres.

(b2.) Montrez, étape par étape, que votre fonction calcule correctement la checksum du nombre "1057", comme dans le premier exemple.

Exemples :

```
1  luhn_checksum("1057")    # returns 0
2  luhn_checksum("12344")   # returns 0
3  luhn_checksum("3456780") # returns 3
```

(c) `is_valid`

Un numéro est *valide* si sa Luhn checksum est égale à 0. Écrivez une fonction `is_valid` qui prend en argument une chaîne `number`, appelle `luhn_checksum` et renvoie `True` si la checksum vaut 0, et `False` sinon.

Vous devez utiliser la fonction `luhn_checksum` à l'intérieur de `is_valid`.

Exemples :

```
1  luhn_checksum("1057")  # returns True
2  is_valid("12344")      # returns True
3  is_valid("3456780")    # returns False
```

(d) `fill_checksums`

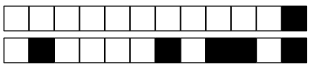
Nous voulons maintenant un second dictionnaire `checksums` qui ait les mêmes clés que `id_numbers`, mais dont les valeurs stockent la checksum et la validité de chaque numéro.

Écrivez une fonction `fill_checksums` qui construit et renvoie un nouveau dictionnaire `checksums` ayant les mêmes clés que `id_numbers`. Pour chaque nom, la valeur associée doit être un couple *checksum, valid* (*un tuple en anglais*) dont la première composante est la checksum et la seconde un booléen (`True` si le numéro est valide, `False` sinon).

Vous devez utiliser les fonctions `luhn_checksum` et `is_valid` à l'intérieur de `fill_checksums`.

Une implémentation correcte de `fill_checksums` doit produire :

```
{
    "Amelie":    (0, True),
    "Benoit":    (3, False),
    "Chloe":     (9, False),
    "Dominique": (0, True),
    "Elodie":    (5, False),
}
```



Question 9: *Cette question est notée sur 5.5 points.*

☐0 ☐.5 ☐1 ☐.5 ☐2 ☐.5 ☐3 ☐.5 ☐4 ☐.5 ☐5 ☐.5

On dit qu'une liste est un *palindrome* si elle se lit de la même façon de gauche à droite et de droite à gauche. Par exemple, la liste `[1, 2, 3, 2, 1]` est un palindrome, alors que `[1, 2, 3, 4]` n'en est pas un.

Vous pouvez supposer que les éléments de la liste peuvent être comparés à l'aide de l'opérateur d'égalité `==` ou de l'inégalité `!=`, et que la liste peut contenir un nombre d'éléments pair ou impair.

Pour chaque sous-question où l'on vous demande d'écrire une fonction, vous devez fournir la définition complète de la fonction, et non seulement le corps. Le non-respect des consignes spécifiques d'implémentation indiquées ci-dessous (par exemple l'utilisation des fonctions imposées ou des constructions demandées) entraînera l'attribution d'au plus 50% des points prévus pour la question concernée, voire moins selon le nombre et la nature des écarts par rapport aux consignes.

(a) `is_palindrome_iter`

(a1.) Écrivez une fonction Python *non récursive* `is_palindrome_iter` qui prend en argument une liste `lst` et renvoie `True` si et seulement si `lst` est un palindrome, et `False` sinon.

Vous pouvez utiliser des boucles, l'indexation et des fonctions prédéfinies simples (par exemple `len`, `range`). En revanche, vous ne devez pas utiliser de mécanisme qui renverse directement la liste, comme *list slicing*, la fonction `reversed`, ou d'autres constructions équivalentes qui comparent `lst` à une version renversée d'elle-même. Vous ne devez pas modifier `lst`.

(a2.) Expliquez en détail comment l'exécution de votre fonction se déroule sur l'exemple concret `lst = ['a', 1, 'b', 1, 'a']`. Décrivez quelles opérations sont effectuées et pourquoi le résultat final est `True` ou `False`.

(a3.) Votre fonction fonctionne-t-elle pour une liste contenant un nombre pair d'éléments ? Justifiez votre réponse.

(b) `is_palindrome_recursive`

(b1.) Écrivez maintenant une fonction Python récursive `is_palindrome_recursive` qui prend en entrée une liste `lst` et un indice `i` (valeur par défaut `0`) correspondant à la position d'un élément de la liste. La fonction doit renvoyer `True` si et seulement si `lst` est un palindrome, et `False` sinon.

En utilisation normale, la fonction sera appelée sous la forme `is_palindrome_recursive(lst)` ; le paramètre `i` est utilisé en interne par la récursion.

Votre solution ne doit pas utiliser de *list slicing*, ni de boucles, et ne doit pas tenter de construire une copie inversée de la liste. Vous ne devez pas modifier `lst`.

(b2.) Expliquez en détail comment l'exécution de votre fonction se déroule sur le même exemple `lst = ['a', 1, 'b', 1, 'a']`. Décrivez chaque appel de la fonction récursive, comment les arguments évoluent, et à quel moment on atteint un cas de base.

Pour votre examen, imprimez de préférence les documents compilés à l'aide de auto-multiple-choice.