Information, Calcul et Communication EPFL – GC/MX – Automne 2021-2022

Examen final : Série A

Cet examen est composé de deux parties.

— La première partie (pages 3 à 6) est composée de 16 questions à choix multiples (8 sur la partie théorique, 8 sur la partie programmation). Pour chaque question, une seule réponse est valable. Un point est attribué pour chaque question répondue correctement. Il n'y a pas de points négatifs.

Pour cette partie, indiquez vos réponses sur le tableau ci-dessous
(et non sur les pages qui suivent !)

Question n°	1	2	3	4	5	6	7	8
Réponse								

Question n°	9	10	11	12	13	14	15	16
Réponse								

— La deuxième partie (pages 7 à 13) est composée de 4 questions ouvertes, valant respectivement 4, 4, 4 et 12 points.

Pour cette partie, répondez sur les feuilles correspondantes !

Le temps total pour l'examen est de 3 h 00.

Documents autorisés : version papier du matériel de cours, vos notes de cours manuscrites, les séries d'exercices et solutions, dictionnaire de langue.

Non autorisés : calculateur, ordinateur, smartphone, tablette ou autre.

Résultat :

QCM 1 (8 pts)	**QCM 2** (8 pts)	**Prob. 1** (4 pts)	**Prob. 2** (4 pts)	**Prob. 3** (4 pts)	**Prob. 4** (12 pts)	**Total** (40 pts)

(Laisser en blanc)

PARTIE QCM : N'OUBLIEZ PAS DE REPORTER VOS RÉPONSES SUR LA PREMIÈRE PAGE !

QCM sur la partie théorique.

Question 1. Soit X un signal de bande passante B_X (on suppose de plus que ce signal est une somme finie de sinusoïdes). Sans rien connaître de plus sur le signal X, que peut-on dire sur la bande passante B_Y du signal Y défini par

$$Y(t) = (X(t))^3, \quad t \in \mathbb{R} \, ?$$

(A) $B_Y = B_X$

(B) $B_Y \leq (B_X)^3$

(C) $B_Y \leq 3\, B_X$

(D) Si on n'a pas plus d'informations sur le signal X, on ne peut rien dire à propos de B_Y.

Question 2. On considère le signal X défini par :

$$X(t) = \sin(4\,\pi^2\, t) + \sin(8\,\pi^2\, t) \quad t \in \mathbb{R}$$

On filtre d'abord ce signal avec un filtre passe-bas idéal de fréquence de coupure f_c, puis on l'échantillonne avec une fréquence d'échantillonnage f_e et à partir de ces échantillons, on reconstruit finalement le signal avec la formule d'interpolation vue au cours (en utilisant la fonction sinc). Appelons $Y = (Y(t),\, t \in \mathbb{R})$ le signal ainsi reconstruit. Laquelle des affirmations suivantes est vraie ?

(A) $Y(t) = X(t)$ pour tout $t \in \mathbb{R}$ si $f_c = 7$ Hz et $f_e = 15$ Hz

(B) $Y(t) = 0$ pour tout $t \in \mathbb{R}$ si $f_c = 7$ Hz et $f_e = 15$ Hz

(C) $\exists t \in \mathbb{R}$ tel que $Y(t) \neq X(t)$ si $f_c = 16$ Hz et $f_e = 30$ Hz

(D) Aucune des affirmations ci-dessus n'est vraie.

Question 3. Parmi les séquences de lettres suivantes, laquelle a la plus petite entropie ?

(A) ABC (B) ABCC (C) ABBCC (D) AABBCC

Indication : $\log_2(3) \simeq 1.58$, $\log_2(5) \simeq 2.32$

Question 4. On utilise le codage de Huffman pour encoder les séquences suivantes :

(A) ABCD (B) ABCDD (C) ABCDDD (D) ABCDDDD

Pour quelle séquence utilisera-t-on le moins de bits par lettre (en moyenne) ?

Question 5. Alice utilise le dictionnaire suivant pour transmettre un message à Bob :

message	En avant !	Recule !	A gauche !	A droite !	Stop !
mot de code	111000	000111	101010	010101	000000

Ce dictionnaire permet à Bob de :

(A) corriger une erreur ou deux effacements

(B) détecter une erreur ou corriger deux effacements

(C) détecter une erreur ou corriger un effacement

(D) seulement corriger un effacement

Note : Par « erreur », on entend le changement d'un 0 en un 1, ou réciproquement.

Question 6. Si deux nœuds A_1, A_2 utilisent l'algorithme AIMD pour transmettre des paquets à des taux W_1, W_2 (respectivement) à travers internet, alors :

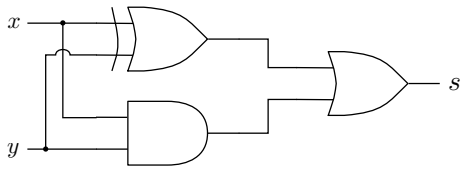
(A) Le rapport W_1/W_2 se rapproche de 1 quand un problème de congestion survient dans le réseau.

(B) Le rapport W_1/W_2 se rapproche de 1 quand il n'y a pas de problème de congestion.

(C) Le rapport W_1/W_2 s'éloigne de 1 quand il n'y a pas de problème de congestion.

(D) Le rapport W_1/W_2 reste identique quand il n'y a pas de problème de congestion.

Question 7. On considère le circuit logique suivant :



Pour des entrées x et y données, que vaut la sortie s de ce circuit ?

(A) $s = x$ OU y (B) $s = x$ ET y (C) s vaut toujours 0 (D) s vaut toujours 1

Question 8. Une attaque par force brute prend 1 heure de temps pour déchiffrer une clé composée de 20 lettres minuscules et majuscules. Combien de temps prendra cette même attaque (avec un ordinateur possédant la même puissance de calcul) pour déchiffrer une clé composée de 20 lettres minuscules uniquement ?

(A) une demi-heure (B) environ 3 minutes

(C) environ 3 secondes (D) environ 3 millisecondes

QCM sur la partie programmation.

Question 9. Qu'affiche cet extrait de code ?

```
d: Dict[int, str] = {}
d[80] = "huitante"
d[80] = "quatre-vingts"
print(f"{len(d)}, {d[80]}")
```

(A) `2, huitante` (B) `2, quatre-vingts` (C) `1, quatre-vingts` (D) Une erreur

Question 10. Qu'affiche cet extrait de code ?

```
words = "fort belle elle dort".split(" ")
transformed_items = [word[1:] for word in words]
print(len(set(transformed_items)))
```

(A) `2` (B) `3` (C) `4` (D) Une erreur

Question 11. Qu'affiche cet extrait de code ?

```
values = [[i for i in range(2, j, 2)] for j in range(10, 2, -2)]
print(values)
```

(A) `[[2], [4, 4], [6, 4, 2], [8, 6, 4, 2], [10, 8, 6, 4, 2]]`

(B) `[[2, 4, 6, 8], [2, 4, 6], [2, 4], [2]]`

(C) `[[10, 8, 6, 4, 2], [10, 8, 6, 4], [10, 8, 6], [10, 8], [10]]`

(D) Une erreur

Question 12. Considérez cet extrait de code, qui crée une grille de 3×3 boutons dans une fenêtre Tkinter. Combien de boutons seront redimensionnés horirontalement (donc élargis) si on élargit la fenêtre ainsi créée ?

```
root = Tk()
root.rowconfigure(1, weight=1)
root.rowconfigure(2, weight=1)
root.columnconfigure(2, weight=2)
for r in range(3):
    for c in range(3):
        but = Button(root, text=f"{r},{c}")
        but.grid(row=r, column=c, sticky=NSEW)
root.mainloop()
```

(A) `1` (B) `2` (C) `3` (D) `6`

Question 13. Quelle affirmation sur le problème des données partagées dans un contexte de programmation concurrente n'est **pas** correcte ?

(A) Plusieurs threads peuvent lire les mêmes données en mémoire sans lock et sans créer de problème de cohérence des données.

(B) Plusieurs threads peuvent modifier les mêmes données en mémoire sans lock et sans créer de problème de cohérence des données.

(C) Les locks peuvent empêcher plusieurs threads de lire les mêmes données en mémoire.

(D) Les locks peuvent empêcher plusieurs threads de modifier les mêmes données en mémoire.

Question 14. Qu'affiche cet extrait de code ?

```
T = TypeVar("T")
def reduce(a: T, b: T, c: T, f: Callable[[T, T], T]) -> T:
    return f(a, f(b, c))
def sub(a: int, b: int) -> int:
    return a - b
print(reduce(1, 2, 3, sub))
```

(A) `2` (B) `-4` (C) `-2` (D) `-1`

Question 15. Imaginons que la classe `Fruit` est une superclasse de la classe `Banana` selon l'extrait de code ci-dessous. Quelle affirmation n'est **pas** correcte ?

```
@dataclass
class Fruit:
    ...
@dataclass
class Banana(Fruit):
    ...
```

(A) Si l'on a une variable de type `List[Fruit]`, la liste pourra également contenir des éléments de type `Banana`.

(B) On a le droit de créer des objets de `Fruit` *et* des objets de type `Banana`.

(C) Dans l'implémentation de la classe `Banana`, on pourra choisir quels champs et quelles méthodes de `Fruit` on souhaite importer et quels autres on souhaite ignorer.

(D) La classe `Fruit`, étant la superclasse de `Banana`, pourrait elle-même hériter d'une autre classe.

Question 16. Que fait cet extrait de code avec les lignes du fichier `data.txt` ?

```
with open("data.txt", "r", encoding="utf8") as file:
    for line in file.read().split("\n"):
        print(line.upper()[::-1])
```

(A) Afficher seulement les lignes qui contiennent des majuscules, dans l'ordre inverse.

(B) Afficher en majuscule le dernier caractère de chaque ligne.

(C) Afficher chaque ligne, tout en majuscules, du dernier au premier caractère.

(D) Afficher la dernière ligne tout en majuscules.

Problème 1. (4 points)

Un signal sonore $X = (X(t),\, t \in \mathbb{R})$ de bande passante $B = 6$ kHz est d'abord filtré par un filtre passe-bas idéal de fréquence de coupure f_c avant d'être échantillonné à une fréquence d'échantillonnage f_e, et chaque échantillon est ensuite encodé sur 32 bits.

a) Supposons que l'on souhaite enregistrer 50 minutes de ce signal sur un support mémoire d'une taille de 120 Mégaoctets (un octet représentant 8 bits). Quelles fréquences f_c et f_e choisir pour conserver au maximum les fréquences du signal X sans pour autant subir l'effet stroboscopique lors de la reconstruction du signal ?

Votre réponse :

b) Si maintenant le signal X est donné par la formule explicite suivante :

$$X(t) = \sum_{n \geq 1} \frac{1}{n} \sin\left(\frac{2\pi f_0 t}{n}\right), \quad t \in \mathbb{R}$$

où $f_0 = 6$ kHz, quelle sera la formule pour le signal reconstruit Y dans le scénario précédent ?

Votre réponse :

Résultat :

a) (2 pts)	**b)** (2 pts)	**Total** (4 pts)

(Laisser en blanc)

Problème 2. (4 points)

Un texte est composé pour moitié de lettres tirées d'un alphabet de 16 lettres, et pour moitié d'espaces. On suppose de plus que les probabilités d'apparition de toutes les lettres sont égales, que les lettres et les espaces sont mélangés dans le texte, et que plusieurs espaces peuvent être consécutifs.

a) Proposez un système d'encodage de ce texte sous la forme d'une séquence de bits qui permette d'utiliser le plus petit nombre moyen de bits par caractère (lettre ou espace), tout en restant décodable de manière unique au moyen d'un dictionnaire ; que vaut ce nombre ?

Votre réponse :

b) Si maintenant le texte était composé pour deux tiers de lettres et pour un tiers d'espaces, l'encodage proposé en a) serait-il toujours optimal ? Calculez également le nombre moyen de bits utilisés par caractère dans ce cas.

Votre réponse :

Résultat :

a) (2 pts)	**b)** (2 pts)	**Total** (4 pts)

(Laisser en blanc)

Problème 3. (4 points)

Afin d'envoyer un message sur internet, Alice décompose d'abord celui-ci en deux paquets de données X_1, X_2, composés chacun d'une séquence de 1024 bits. Alice envoie ensuite X_1 et X_2, ainsi que deux copies $X_3 = X_1$ et $X_4 = X_2$ de ces paquets, et également un cinquième paquet $X_5 = X_1 \oplus X_2$ (ce qui signifie que chaque bit de X_5 est l'addition modulo 2 des bits des paquets X_1 et X_2).

a) Bob est le destinataire de ces paquets, mais malheureusement, certains d'entre eux se perdent en cours de route... Combien de pertes de paquets Bob peut-il tolérer, dans le pire des cas, tout en restant sûr de pouvoir retrouver le message orignal X_1, X_2 envoyé par Alice ? Expliquez votre raisonnement.

Votre réponse :

b) Supposons maintenant qu'Alice et Bob se mettent d'accord à l'avance sur une clé secrète K, dont les bits sont tirés uniformément au hasard. Expliquez comment Alice et Bob peuvent utiliser cette clé pour échanger les cinq paquets décrits ci-dessus, de sorte que si Eve intercepte même tous les paquets, elle ne puisse retrouver aucune information à propos de X_1 ou X_2. En particulier, quelle doit être la longueur minimum de la clé K (en nombre de bits) ?

Votre réponse :

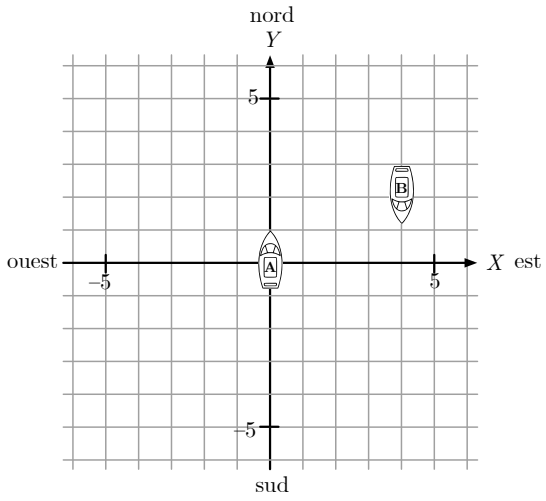
Résultat :

a) (2 pts)	**b)** (2 pts)	**Total** (4 pts)

(Laisser en blanc)

Problème 4. (12 points)

On décide de modéliser les mouvements d'un petit bateau robot dans un le plan avec une classe Python. Un tel bateau se déplace sur une grille comme illustré ci-dessous et a toujours des coordonnées x et y entières. Il peut être orienté vers le nord, l'est, le sud ou l'ouest. Son état initial est toujours en position $(0, 0)$, orienté vers le nord, comme montré ici avec le bateau **A** :



Le bateau **B**, dans le premier quadrant, est ici en position $(4, 2)$ et est orienté vers le sud.

a) Écrivez une **`dataclass`** pour modéliser la position et l'orientation du bateau. Pour chacun des champs que vous déclarez, indiquez (en commentaires) quelles valeurs ce champ pourra prendre selon la modélisation que vous choisissez.

```
@dataclass
class Boat:
```

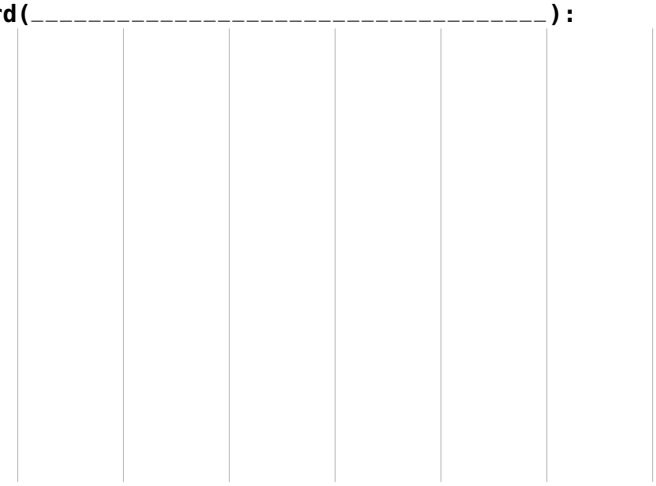
Le bateau sait faire deux types de mouvement : (1) avancer d'une unité dans la direction dans laquelle il est orienté ; (2) pivoter sur place de 90 degrés dans le sens des aiguilles d'une montre ou dans le sens contraire.

b) Sur la page suivante, complétez la classe **`Boat`** en ajoutant deux méthodes : (1) la première, **`forward`**, doit faire avancer le bateau d'un nombre entier d'unités passé en paramètre ; (2) la seconde, **`turn`**, doit le faire pivoter de 90 degrés dans le sens choisi par un paramètre booléen appelé **`clockwise`**. Indiquez également les types des paramètres.

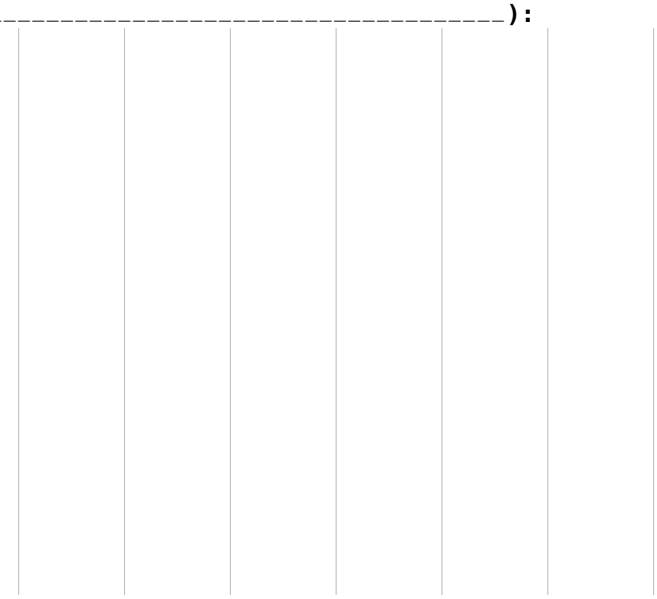
Complétez l'affection de la variable **`boat`** pour créer un bateau dans son état initial. Veillez à ce que vos méthodes soient définies de manière telle que les appels des lignes suivantes soient possibles et emmènent le bateau dans la position **B** du premier quadrant montrée dans la figure ci-dessus.

```
@dataclass
class Boat:
        # (les champs comme ci-dessus, puis...)

        def forward(_____________________________________):
```



```
        def turn(_____________________________________):
```



```
boat = _________________________________
boat.forward(3)
boat.turn(clockwise=True)
boat.forward(4)
boat.turn(clockwise=True)
boat.forward(1)
```

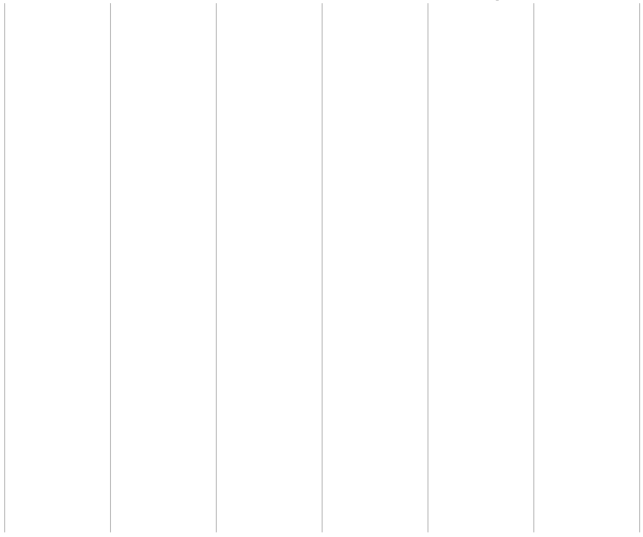
On cherche maintenant à piloter le bateau en lui donnant une série d'instructions, représentées par une chaîne de caractères dans laquelle chaque caractère est :
— soit `"F"` pour représenter une avancée d'une unité ;
— soit `"R"` pour représenter un pivotement dans le sens des aiguilles d'une montre ;
— soit `"L"` pour représenter un pivotement dans le sens contraire.

Par exemple, cette chaîne de caractères doit effectuer les mêmes mouvements que le code de l'extrait ci-dessus : **`"FFFRFFFFRF"`**.

c) Ajoutez à la classe **`Boat`** une méthode **`exec`** qui prend en paramètre une telle chaîne de caractères et exécute les mouvements spécifiés par les caractères individuels, en ignorant les éventuels caractères non valides.

```
@dataclass
class Boat:
        # (les champs et méthode comme ci-dessus, puis...)

        def exec(_____________________________________):
```



d) Écrivez une fonction **`are_paths_equivalent`** (donc pas une méthode de la classe **`Boat`**) qui accepte deux chaînes d'instructions et qui renvoie **`True`** si les deux chaînes mènent le bateau au même endroit et que l'orientation finale est la même, et qui renvoie **`False`** sinon. Indiquez aussi son type de retour. Pour l'implémenter, vous pouvez faire appel à ce qui a été défini plus haut comme champs et méthodes ainsi qu'implémenter des fonctions auxiliaires — si cela vous aide, bien sûr.

```
def are_paths_equivalent(_____________________________________________) -> ____________:
```

e) Parfois, une chaîne d'instructions fait plus de détours que nécessaire pour arriver à sa destination. Écrivez une fonction **`optimize`** (aussi en dehors de la classe **`Boat`**) qui accepte une chaîne d'instructions et qui renvoie la chaîne d'instruction la plus courte possible qui permette au bateau d'arriver à la même position (sans se préoccuper de son orientation finale). Par exemple :
— l'appel **`optimize("FFFRFFFFRF")`** doit renvoyer **`"FFRFFFF"`** ;
— l'appel **`optimize("FLFLFLFLLRL")`** doit renvoyer **`""`** (aucun déplacement n'est nécessaire) ;
— l'appel **`optimize("RRFFF")`** peut renvoyer soit **`"RRFFF"`**, soit **`"LLFFF"`**.

Comme pour d), vous pouvez faire appel à du code défini plus haut et implémenter des fonctions auxiliaires. Comme indice, rappelez-vous que l'expression **`"x" * 3`** donne comme valeur **`"xxx"`**.

```
def optimize(__________________________) -> ____________:
```

Résultat :

a) (1 pt)	**b)** (3 pts)	**c)** (2 pts)	**d)** (2 pts)	**e)** (4 pts)	**Total** (12 pts)

(Laisser en blanc)

Cette page sert d'espace supplémentaire pour répondre aux questions des pages précédentes.