Corrected

EPFL

**Enseignant : Rafael Pires**
**CS-119(k) ICC Final - CS**
**27.06.2025 9h15**
**180 min.**

1

# Tartempion Machinchose

SCIPER : **000000** Salle : **PO 01** Signature :

**Attendez le début de l'épreuve avant de tourner la page. Ce document est imprimé recto-verso, il contient 16 pages, les dernières pouvant être vides. Ne pas dégrafer.**

- Posez votre carte d'étudiant sur la table.
- **Aucun** document n'est autorisé.
- L'utilisation d'une **calculatrice** et de tout outil électronique est interdite pendant l'épreuve.
- Première partie :
  - Pour les questions **avec une seule bonne réponse possible**, on comptera,:

    +3 points si la réponse est correcte,
    0 point si il n'y a aucune ou plus d'une réponse inscrite,
    −1 point si la réponse est incorrecte.

  - Pour les questions **avec plusieurs bonnes réponses possibles**, **pour chaque choix de réponse**, on comptera,:
    * ☒ Cette réponse est correcte : si cochée, +1 point, sinon -1 point
    * ☐ Cette réponse n'est **pas** correcte : si cochée, -1 point, sinon +1 point
    * Si pas de réponse **à la question**, 0 point
- Deuxième partie : Pour les questions **vrai/faux**, on comptera,:

  +1 points si la réponse est correcte,
  0 point si il n'y a aucune ou plus d'une réponse inscrite,
  −1 point si la réponse est incorrecte.
- Troisième partie : pour les questions ouvertes, le nombre de point maximum est noté au-dessus de chaque question. Laissez les cases à cocher vides !
- Utilisez un **stylo** à encre **noire ou bleu foncé** et effacez proprement avec du **correcteur blanc** si nécessaire.
- Si une question est erronée, l'enseignant se réserve le droit de l'annuler.

Respectez les consignes suivantes | Observe this guidelines | Beachten Sie bitte die unten stehenden Richtlinien

| choisir une réponse \| select an answer<br>Antwort auswählen | ne PAS choisir une réponse \| NOT select an answer<br>NICHT Antwort auswählen | Corriger une réponse \| Correct an answer<br>Antwort korrigieren |
|---|---|---|
| ☒ ☑ ■ | ☐ | ☐ |

ce qu'il ne faut **PAS** faire | what should **NOT** be done | was man **NICHT** tun sollte

CORRECTED

## Première partie, questions à choix multiple

Pour chaque question marquer la/les case(s) correspondante(s) à la/aux réponse(s) correcte(s) sans faire de ratures.

**Question 1** Que faut-il faire pour éviter l'effet stroboscopique ?

- [x] Appliquer un filtre passe-bas idéal avant l'échantillonage
- [x] Augmenter suffisament la fréquence d'échantillonage
- [ ] Réduire suffisament la fréquence d'échantillonage
- [ ] Appliquer un filtre passe-haut avant l'échantillonnage

**Question 2** On considère le signal suivant :

$$x(t) = 3\sin\left(\frac{10\pi}{3}t\right) + 2\sin(6\pi t) + \sin(5\pi t + \frac{\pi}{6})$$

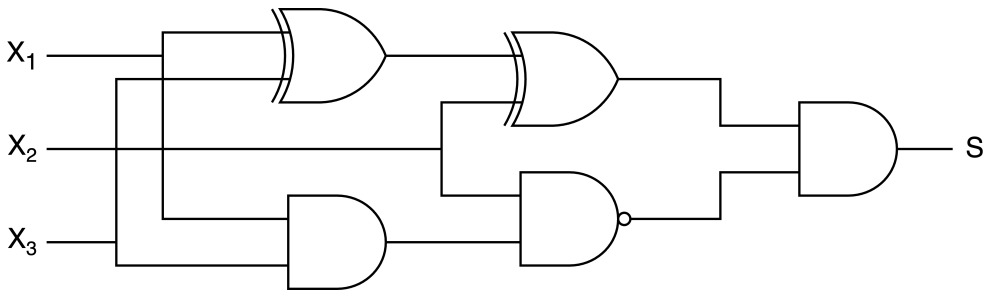
On souhaite échantillonner ce signal à une fréquence $f_e$ (en Hz), puis le reconstruire **parfaitement** par interpolation sinc.
Quelle(s) est(sont) la(les) fréquence(s) d'échantillonnage **admissible(s)** parmi les suivantes ?

- [ ] 6 Hz
- [x] 8 Hz
- [x] 10 Hz
- [ ] 3 Hz

**Question 3** Pour que la sécurité de mon one-time pad soit parfaite alors (cochez l'(les) option(s) correcte(s)) :

- [ ] La même clé peut être réutilisée pour plusieurs messages tant qu'elle est plus grande que les messages.
- [ ] La clé est parfaitement aléatoire.
- [ ] Un algorithme de suffisament grande complexité est utilisé.
- [x] La clé a au moins la même longueur que le message.

**Question 4** On considère le circuit suivant :




Quelle(s) proposition(s) est(sont) correcte(s) (Pour rappel, $\oplus$ est l'opération XOR vue dans le cours)?

- [x] $s = 1$ seulement quand la somme des entrées est égale à 1
- [x] $s = x_1 \oplus x_2 \oplus x_3$
- [ ] $s = x_1 + x_2 + x_3$
- [ ] $s = 1$ seulement quand la somme des entrées est égale à 2

CORRECTED

**Question 5** Parmi les affirmations suivantes concernant les classes de complexité P et NP, cochez celles qui sont **vraies**.

- ☐ Tous les problèmes NP sont aussi dans P.
- ☐ Les problèmes NP-complets peuvent être résolus en temps polynomial.
- ■ La classe P contient les problèmes décidables en temps polynomial.
- ☐ La classe NP contient les problèmes pour lesquels une solution ne peut pas être vérifiée en temps polynomial.

**Question 6** Parmi les propositions suivantes concernant le hachage (stockage de mots de passe), les signatures numériques et la cryptographie asymétrique, cochez *toutes* celles qui sont correctes.

- ■ Une signature numérique se crée avec la clé privée du signataire et se vérifie à l'aide de sa clé publique.
- ☐ Pour assurer la confidentialité d'un message, on le chiffre avec la clé privée du destinataire.
- ■ Ajouter un sel aléatoire propre à chaque utilisateur (`hash(salt + mot_de_passe)`) empêche qu'on puisse déduire qu'un même mot de passe est utilisé par plusieurs comptes.
- ☐ Le sel utilisé pour le hachage des mots de passe doit rester secret et ne jamais être stocké en clair.

**Question 7** Quelle est la bande passante du signal suivant:

$$X(t) = \mathrm{sinc}(2t) = \frac{\sin(2\pi t)}{2\pi t}$$

- ☐ Infinie
- ☐ 2 Hz
- ■ 1 Hz
- ☐ $\frac{1}{2}$ Hz

**Question 8** On considère le signal suivant

$$X(t) = 4\sin(6\pi t)\cos(3\pi t)\sin(2\pi t).$$

Quelle est la bande passante de ce signal ?

On rappelle que :

$$2\sin(u)\sin(v) = \cos(u-v) - \cos(u+v) \text{ et } \cos(u)\sin(v) = \sin(u+v) - \sin(u-v)$$

On rappelle également que :

$$\cos(a) = \sin(a + \tfrac{\pi}{2}),\ \cos(a) = \sin(\tfrac{\pi}{2} - a), \cos(-a) = \cos(a) \text{ et } \sin(-a) = -\sin(a)$$

- ☐ 3 Hz
- ☐ 11 Hz
- ■ 5.5 Hz
- ☐ 6 Hz

CORRECTED

**Question 9** Soient $x = 100$ et $y = 60$ deux nombres entiers positifs, chacun représenté en binaire sur 8 bits. On effectue une opération bit-à-bit NOT($x$ AND $y$). On obtient ainsi une nouvelle séquence de 8 bits. Quel est le nombre entier positif $z$ représenté par cette nouvelle séquence de 8 bits ?

- [x] 219
- [ ] 24
- [ ] 36
- [ ] 131

**Question 10** On considère l'algorithme suivant :

**Algorithm 1:** algo_mystère

**Input:** nombre entier strictement positif $n$
**Output:** ???

**if** $n = 1$ **or** $n = 2$ **then**
  **return** 1
$s \leftarrow 0$
**for** $i \leftarrow 1$ **to** $n - 2$ **do**
  $s \leftarrow s +$ algo_mystère($i$)
**return** $s$

Pour une valeur de $n \geq 3$ donnée, que vaut la sortie de cet algorithme ?

- [ ] $2^{n-2}$
- [ ] $2^{n-3}$
- [ ] $F(n-2)$
- [x] $F(n-1)$

CORRECTED

**Question 11** Une source $X$ émet les symboles suivants :

$$\{A, B, C, D\}$$

avec les probabilités respectives :

$$\left\{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}\right\}$$

On note :

- $H(X)$ l'entropie de la source
- $L_H(X)$ la longueur moyenne du code de Huffman
- $L_{\mathrm{SF}}(X)$ la longueur moyenne du code Shannon–Fano

Quelle est la valeur de $L_H(X)$ ?

- ☐ 2.0
- ■ 1.75
- ☐ 1.5
- ☐ 1.625

Quelle relation est correcte ?

- ☐ $H(X) = L_H(X) < L_{\mathrm{SF}}(X)$
- ☐ $H(X) < L_H(X) < L_{\mathrm{SF}}(X)$
- ■ $H(X) = L_H(X) = L_{\mathrm{SF}}(X)$
- ☐ $H(X) < L_H(X) = L_{\mathrm{SF}}(X)$

**Question 12** Considérez l'algorithme récursif suivant :

```python
def mystere(tab):
    if len(tab) <= 1:
        return tab
    mid = len(tab) // 2
    left = mystere(tab[:mid])
    right = mystere(tab[mid:])
    return fusion(left, right)
```

On suppose que la fonction `fusion` combine deux tableaux triés de taille $n_1$ et $n_2$ en temps linéaire $\mathcal{O}(n_1+n_2)$. Quelle est la complexité en pire cas de l'algorithme `mystere` en fonction de $n = $ `len(tab)`?

- ☐ $\mathcal{O}(n)$
- ■ $\mathcal{O}(n \log n)$
- ☐ $\mathcal{O}(n^2)$
- ☐ $\mathcal{O}(2^n)$

CORRECTED

## Deuxième partie, questions du type Vrai ou Faux

Pour chaque question, marquer (sans faire de ratures) la case VRAI si l'affirmation est **toujours vraie** ou la case FAUX si elle **n'est pas toujours vraie** (c'est-à-dire si elle est parfois fausse).

**Question 13** En Python, la construction `with open(... ) as file:` assure la fermeture automatique (`close()`) du fichier à la fin du bloc, même si une exception est levée.

■ VRAI ☐ FAUX

**Question 14** En Python, la syntaxe des compréhensions de liste permet d'enchaîner plusieurs clauses `for` dans une même expression.

■ VRAI ☐ FAUX

**Question 15** Lorsque plusieurs threads modifient simultanément une même variable partagée sans mécanisme de verrou, on n'arrive pas à une condition de course (*race condition*).

☐ VRAI ■ FAUX

**Question 16** Un algorithme correspond à l'implémentation d'un programme dans un langage donné.

☐ VRAI ■ FAUX

**Question 17** L'algorithme « Tous différents » vu en cours réalise $\Theta(n^2)$ comparaisons sur une liste non triée dans le pire des cas.

■ VRAI ☐ FAUX

**Question 18** Tout algorithme récursif doit comporter une condition de terminaison.

■ VRAI ☐ FAUX

**Question 19** La mémoïsation consiste à stocker les résultats intermédiaires afin d'éviter les recalculs inutiles.

■ VRAI ☐ FAUX

**Question 20** L'ensemble de tous les algorithmes (quel que soit le formalisme) est *dénombrable*.

■ VRAI ☐ FAUX

**Question 21** Le problème de l'arrêt (*Halting Problem*) est décidable : il existe un algorithme capable de déterminer, pour tout programme et toute entrée, si l'exécution s'achèvera ou non.

☐ VRAI ■ FAUX

**Question 22** Avec 8 bits en complément à deux, on peut représenter les entiers de $-127$ à $+128$.

☐ VRAI ■ FAUX

**Question 23** En électronique, on peut réaliser toute fonction logique à l'aide d'un réseau composé uniquement de portes NAND.

■ VRAI ☐ FAUX

**Question 24** Un filtre passe-bas idéal laisse passer toutes les composantes de fréquences supérieures à sa fréquence de coupure.

☐ VRAI ■ FAUX

Corrected

**Question 25** D'après le théorème d'échantillonnage de Nyquist–Shannon, si la bande passante maximale d'un signal vaut $f_{\max}$, une fréquence d'échantillonnage $f_s \geq 2f_{\max}$ est suffisante pour reconstruire exactement le signal.

☐ VRAI ■ FAUX

**Question 26** Pour une source utilisant un alphabet de taille $n$, l'entropie $H(X)$ vérifie $0 \leq H(X) \leq \log_2 n$.

■ VRAI ☐ FAUX

**Question 27** La sécurité du protocole d'échange de clé Diffie–Hellman repose sur la difficulté de factoriser de grands nombres entiers.

☐ VRAI ■ FAUX

**Question 28** Dans l'architecture TCP/IP, c'est la couche Transport qui décide du chemin pris par les paquets à travers le réseau.

☐ VRAI ■ FAUX

CORRECTED

## Troisième partie, questions de type ouvert

Répondre dans l'espace dédié. Votre réponse doit être soigneusement justifiée, toutes les étapes de votre raisonnement doivent figurer dans votre réponse. Laisser libres les cases à cocher : elles sont réservées au correcteur.

**Question 29:** *Cette question est notée sur 8 points.*

☐ 0 ☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 ■ 8

On considère la séquence de 36 bits suivante :

000001010100 000010001100 000100010001

(les espaces ci-dessus sont juste ajoutés pour la lisibilité de la séquence, mais sont à ignorer)
Pour compresser cette séquence, on considère deux stratégies possibles :

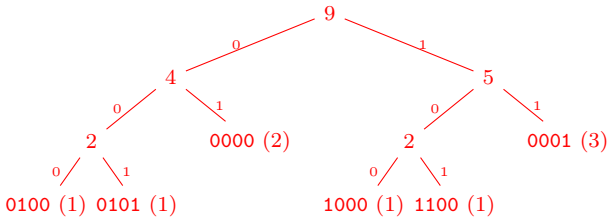
(a) Regrouper les bits par groupes de 4, en considérant chaque groupe de 4 bits comme un seul symbole, et appliquer l'encodage de Huffman à ces symboles.

(b) Regrouper les bits par groupes de 3, en considérant chaque groupe de 3 bits comme un seul symbole, et appliquer l'encodage de Huffman à ces symboles.

Établir le code de Huffman dans chacun des deux cas, et déterminer quel code mène à une meilleure compression de la séquence d'origine.
**Stratégie 1 – groupes de 4 bits (9 symboles).** Découpage : 0000 0101 0100 0000 1000 1100 0001 0001 0001.

Fréquences : 0001→3, 0000→2, et 0100, 0101, 1000, 1100 →1 chacun.

Arbre de Huffman (fusions successives des deux plus petits) :




Codes : 0000→01 (2 bits ×2), 0001→11 (2 bits ×3), 0100→000, 0101→001, 1000→100, 1100→101 (3 bits ×1 chacun).

Longueur totale : $2 \cdot 2 + 3 \cdot 2 + 4 \cdot 3 = 4 + 6 + 12 = \mathbf{22}$ bits.
(Vérification : somme des poids des nœuds internes $= 2 + 2 + 4 + 5 + 9 = 22$.)

**Stratégie 2 – groupes de 3 bits (12 symboles).** Découpage : 000 001 010 100 000 010 001 100 000 100 010 001. Chaque symbole apparaît 3 fois, donc Huffman attribue 2 bits à chacun.

Longueur totale : $12 \times 2 = \mathbf{24}$ bits.

**Conclusion.** La stratégie en **groupes de 4 bits** (22 bits) compresse mieux que celle en groupes de 3 bits (24 bits).

**Question 30:** *Cette question est notée sur 4 points.*

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

**[3 points]** En utilisant RSA, vous souhaitez protéger une information que vous envoyez à un ami dont la clé publique est (17, 77) et la clé privée est $d = 41$. Votre clé publique est (11, 65) et votre clé privée est $d = 23$.

CORRECTED

En utilisant les lettres $m$ et $c$ pour le message en clair et chiffré respectivement, expliquez précisément chaque étape de la communication (en partant du message en clair jusqu'au moment où votre ami récupère le message en clair.

On chiffre avec la **clé publique de l'ami**, et l'ami déchiffre avec sa **clé privée** :
1. **Chiffrement** (chez vous) : $c = m^{e_A} \bmod n_A = m^{17} \bmod 77$.
2. **Transmission** de $c$ sur le canal (un attaquant interceptant $c$ ne peut pas retrouver $m$ sans $d_A$).
3. **Déchiffrement** (chez l'ami) : $m = c^{d_A} \bmod n_A = c^{41} \bmod 77$.

Cela fonctionne car $e_A \cdot d_A \equiv 1 \pmod{\varphi(n_A)}$, donc $m^{e_A d_A} \equiv m \pmod{n_A}$.

**[1 points]** Vous souhaitez envoyer le message $m = 19$ à votre ami. Vous le chiffrez puis l'envoyez. Votre ami déchiffre le message reçu et obtient la valeur 12.

**Note** : quelques valeurs de $a^b \mod c$ (classées par $a$, $b$, $c$) :

| | | | |
|---|---|---|---|
| $19^{11} = 34 \mod 65$ | $19^{17} = 34 \mod 77$ | $19^{23} = 19 \mod 33$ | $19^{23} = 34 \mod 65$ |
| $19^{29} = 58 \mod 65$ | $19^{41} = 12 \mod 77$ | $34^{11} = 19 \mod 77$ | $34^{17} = 12 \mod 77$ |
| $34^{23} = 11 \mod 65$ | $34^{29} = 58 \mod 65$ | $34^{41} = 19 \mod 77$ | $34^{23} = 19 \mod 65$ |

Donnez une possible erreur qui entrainerait cette conséquence. Justifiez votre réponse.

Avec un protocole correct : $c = 19^{17} \bmod 77 = 34$ et $34^{41} \bmod 77 = 19$ ✓. Or l'ami obtient 12.

**Erreur possible** : *vous avez oublié de chiffrer le message et envoyé $m = 19$ en clair.* L'ami, croyant recevoir un message chiffré, lui applique sa clé privée : $19^{41} \bmod 77 = 12$ (table) ✓, ce qui correspond au résultat observé.

*Alternative également cohérente :* l'ami a utilisé sa clé *publique* $e_A = 17$ au lieu de sa clé privée. Sur le chiffré correct $c = 34$ on aurait $34^{17} \bmod 77 = 12$.

**Question 31:** *Cette question est notée sur 4 points.*


**[1 point]** Ajoutez, à la classe `SaleRecord`, une méthode `total_sold()` (indentation correcte indispensable) qui retourne la somme des quantités vendues pour ce produit. Si la liste `daily_quantities` est vide, la méthode doit retourner `-1`.

```python
from dataclasses import dataclass
from typing import List

@dataclass
class SaleRecord:
    product_name: str
    daily_quantities: List[int]

    # TODO: implementer total_sold()

@dataclass
class Store:
    id: str
    city: str
    sales: List[SaleRecord]

# jeu d'essai minimal
store1 = Store(id="ST01", city="Lausanne", sales=[
    SaleRecord("Bananas", [15, 12, 18]),
    SaleRecord("Apples",  [20, 17, 19]),
])
```

```
store2 = Store(id="ST02", city="Geneve",   sales=[
    SaleRecord("Pears",   [ 7,  9,  8]),
])
all_stores: List[Store] = [store1, store2]
```

```
    def total_sold(self) -> int:
        if not self.daily_quantities:
            return -1
        return sum(self.daily_quantities)
```

Le test `not self.daily_quantities` peut aussi s'écrire `len(self.daily_quantities) == 0` (ou `self.daily_quantities == []`).

**[2 points]** Implémentez la fonction suivante. Elle reçoit la liste de tous les magasins, l'identifiant d'un magasin et le nom d'un produit ; elle renvoie le total vendu pour ce produit dans ce magasin (grâce à `total_sold()`). Si le magasin ou le produit n'existe pas, la fonction renvoie `-1`.

```
def get_total_sold(all_stores: List[Store],
                   store_id: str,
                   product_name: str) -> int:
    ...
```

```
def get_total_sold(all_stores: List[Store],
                   store_id: str,
                   product_name: str) -> int:
    for store in all_stores:
        if store.id == store_id:
            for sale in store.sales:
                if sale.product_name == product_name:
                    return sale.total_sold()
            return -1  # produit absent dans ce magasin
    return -1          # magasin inexistant
```

Vérification : `get_total_sold(all_stores, "ST01", "Apples")` renvoie $20 + 17 + 19 = 56$.

**[1 point]** Complétez la variable `sales_index` pour qu'elle relie chaque identifiant de magasin à un dictionnaire « produit → total vendu ». Exemple attendu : `sales_index["ST01"]["Apples"]` vaut `56`.

```
sales_index: dict[str, dict[str, int]] = {}
# votre code ici
```

```
sales_index: dict[str, dict[str, int]] = {}
for store in all_stores:
    sales_index[store.id] = {
        sale.product_name: sale.total_sold()
        for sale in store.sales
    }
```

Variante sans compréhension de dictionnaire (deux boucles imbriquées) :

```
sales_index: dict[str, dict[str, int]] = {}
for store in all_stores:
    sales_index[store.id] = {}
    for sale in store.sales:
        sales_index[store.id][sale.product_name] = sale.total_sold()
```

Résultat : `sales_index == {"ST01": {"Bananas": 45, "Apples": 56}, "ST02": {"Pears": 24}}`.