

ⵜⴰⴳⴷⴰⵢⵜ ⵏ ⵎⴻⵔⴽⵓ
ⵜⴰⴷⵓⵏⵜ ⵏ ⵓⵎⵓⵏⵏⴰ ⵏ ⵉⵎⵎⴰⵔ
ⵓⵙⵙⴰⵏⵜ ⵏ ⵉⵎⵎⴰⵔ ⵏ ⵓⵎⵓⵏⵏⴰ



المملكة المغربية
وزارة التعليم العالي
والبحث العلمي والابتكار

Royaume du Maroc
Ministère de l'Enseignement Supérieur,
de la Recherche Scientifique et de l'Innovation



CONCOURS NATIONAL COMMUN

d'Admission dans les Établissements de Formation d'Ingénieurs et
Établissements Assimilés

Session 2026

Épreuve de : **Informatique**

Filières : **PSI**

Date : **15-05-2026** Durée **2 Heures** de **15H00 à 17H00**

Si, au cours de l'épreuve, un candidat repère ce qui lui semble être un point de clarification ou une erreur d'énoncé, il le signale sur sa copie et poursuit sa composition en expliquant les raisons de son initiative.

L'usage de tout document et de tout matériel électronique ou informatique (y compris la calculatrice) est interdit

Cette épreuve comporte **11 pages** au format **A4** en plus de cette page de garde

Les candidats sont informés que la précision des raisonnements algorithmiques ainsi que le soin apporté à la rédaction et à la présentation des copies seront des éléments pris en compte dans la notation. Il convient en particulier de rappeler avec précision les références des questions abordées. Si, au cours de l'épreuve, un candidat repère ce qui peut lui sembler être une erreur d'énoncé, il le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives qu'il est amené à prendre.

Remarques générales :

- ✓ Cette épreuve est composée d'un exercice et de trois parties tous indépendants ;
- ✓ Toutes les instructions et les fonctions demandées seront écrites en Python ;
- ✓ Les questions non traitées peuvent être admises pour aborder les questions ultérieures ;
- ✓ Toute fonction peut être décomposée, si nécessaire, en plusieurs fonctions.

Important : Le candidat doit impérativement commencer par traiter toutes les questions de l'exercice ci-dessous, et écrire les réponses dans les premières pages du cahier de réponses.

Exercice : (4 points)

Nombre palindrome

L'algorithme suivant, retourne la liste des chiffres d'un entier n strictement positif :

fonction chiffres (n) :

$L \leftarrow$ liste vide

tant que $n \neq 0$ *faire*

$r \leftarrow n \bmod 10$ (r est le reste de la division entière de n par 10)

insérer le reste r *au début de la liste* L

$n \leftarrow n/10$

fin tant que

retourner L

Par exemple, pour $n=7495$, la fonction retourne la liste $L = [7, 4, 9, 5]$

Q.1- Écrire la fonction **chiffres (n)** qui reçoit en paramètre un entier strictement positif n , et qui retourne la liste des chiffres de n , en utilisant l'algorithme ci-dessus.

Q.2- Écrire la fonction **valeur (L)** qui reçoit en paramètre la liste L des chiffres d'un entier strictement positif. La fonction retourne la valeur décimale, en utilisant la formule suivante :

$$\sum L[i] * 10^i$$

Exemple : **valeur ([7, 4, 9, 5])** retourne le nombre $5947 = 7*10^0 + 4*10^1 + 9*10^2 + 5*10^3$

Q.3- Déterminer la complexité de la fonction **valeur (L)**, avec justification.

Q.4- Écrire la fonction **renverse (n)** qui reçoit en paramètre un entier strictement positif n , et qui retourne le nombre composé des chiffres de n , dans l'ordre renversé.

Exemple : La fonction **renverse (7495)** retourne le nombre **5947**

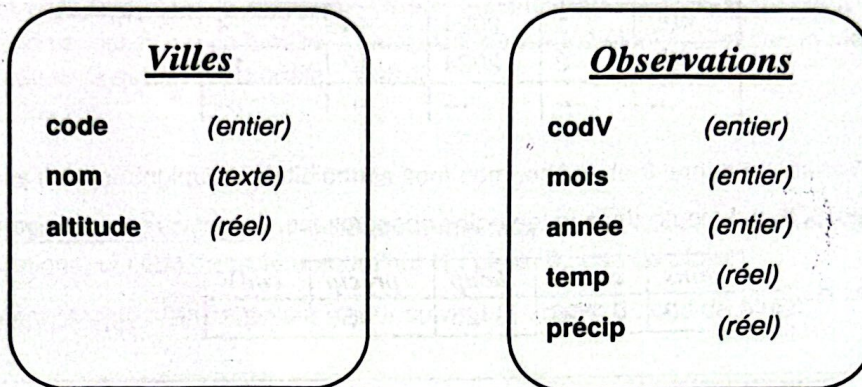
Q.5- Un nombre est un **palindrome** si on peut le lire de gauche à droite comme de droite à gauche. Par exemple, le nombre **47074** est un palindrome.

Écrire la fonction **palindrome (n)** qui reçoit en paramètre un entier strictement positif n , et qui retourne **True** si n est un palindrome, sinon, la fonction retourne **False**.

Q.6- Écrire la fonction **list_palindrome (P)** qui reçoit en paramètre une liste P de nombres entiers strictement positifs. La fonction retourne **True** si tous les nombres de P sont des palindromes, sinon, la fonction retourne **False**.

Partie I : Base de données et langage SQL

Les données provenant des observations météorologiques mensuelles de plusieurs villes, sont enregistrées dans une base de données composée de deux tables appelées : '**Villes**' et '**Observations**'.



Structure de la table 'Villes' :

La table '**Villes**' est composée des champs suivants :

- Le champ **code** contient un code unique pour chaque ville ;
- Le champ **nom** contient le nom de chaque ville ;
- Le champ **altitude** contient l'altitude (en mètre) de chaque ville ;

Exemples :

<i>code</i>	<i>nom</i>	<i>altitude</i>
900	Tanger	56
400	Marrakech	465
200	Casa Blanca	115
100	Rabat	87
...	...	...

Structure de la table 'Observations' :

La table '**Observations**' est composée des champs suivants :

- Le champ **codV** contient le code de la ville ;
- Les champs **mois**, **année** contiennent le mois et l'année de l'observation météorologique dans chaque ville ;
- Le champ **temp** contient la température moyenne (en °C) dans chaque ville, durant un mois ;
- Le champ **précip** contient le cumul (en mm) des précipitations de pluie dans chaque ville, durant un mois.

Exemples :

<i>codV</i>	<i>mois</i>	<i>année</i>	<i>temp</i>	<i>précip</i>
900	1	2024	17	10
900	2	2024	15	14
900	3	2024	16	79
900	1	2025	18	125
400	1	2024	14	4
400	2	2024	15	12
...	...	...	...	...

Q.7- Écrire une requête SQL qui ajoute dans la table des observations, l'enregistrement suivant :

<i>mois</i>	<i>année</i>	<i>temp</i>	<i>précip</i>	<i>codV</i>
3	2025	25	50	400

Q.8- Écrire, en algèbre relationnelle, une requête qui donne pour résultat *les mois, les températures et les précipitations de la ville 'Agadir', durant l'année 2024.*

Q.9- Écrire une requête SQL qui donne pour résultat *les noms et les altitudes des villes, les températures et les précipitations du mois 10 de l'année 2025. Le résultat doit être trié dans l'ordre décroissant des précipitations.*

Q.10- Écrire une requête SQL qui donne pour résultat *le compte des villes qui ont enregistré des températures comprises entre 45 °C et 50 °C, durant le mois 7 ou le mois 8 de l'année 2025.*

Q.11- Écrire une requête SQL qui donne pour résultat *les noms des villes, la plus grande température et la plus petite température par ville durant l'année 2025. Le résultat doit être trié dans l'ordre alphabétique des noms des villes.*

Q.12- Écrire une requête SQL qui donne pour résultat *les années, les noms des villes, le cumul annuel des précipitations, et la moyenne annuelle des températures par ville par année, tels que le cumul annuel des précipitations est inférieur à 450 mm.*

Q.13- Écrire une requête SQL qui donne pour résultat *la moyenne des cumuls annuels des précipitations de chaque ville, durant l'année 2025.*

Partie II :***Va-t-il pleuvoir demain ?***

Dans cette partie, on utilise un ensemble de données provenant des observations météorologiques quotidiennes de plusieurs stations météorologiques. L'objectif est de prédire s'il pleuvra demain ou non, en utilisant le modèle de classification binaire K plus proches voisins (*KNN*). Ensuite, la matrice de confusion est utilisée pour mesurer la qualité du modèle construit.

Les observations météorologiques quotidiennes sont composées de 6 variables, dans l'ordre suivant :

- Cinq variables de prédiction : température minimale, température maximale, température moyenne, précipitations, et l'état de pluie aujourd'hui (1 : pluie, 0 : pas de pluie)
- Une variable cible : l'état de pluie le jour suivant (1 : pluie, 0 : pas de pluie).

température minimale	température maximale	température moyenne	précipitations	l'état de pluie aujourd'hui	état de pluie le jour suivant
22.8	28.9	24.25	1.5	0	0
22.8	29.9	26.0	0.0	0	0
22.4	30.0	24.2	1.1	0	0
22.4	29.7	25.6	11.3	1	0
24.8	30.4	28.1	0.0	0	0
22.0	31.8	25.1	30.2	1	1
...	...	...	...	...	...
22.2	32.5	25.6	0.0	0	0

L'algorithme KNN est un algorithme de Machine Learning qui appartient à la classe des algorithmes d'apprentissage supervisé, utilisé pour résoudre les problèmes de classification ou de régression.

A- Phase d'entraînement du modèle de classification KNN

Dans la phase d'entraînement du modèle, on utilise une liste *E* qui contient un échantillon de 30% des observations météorologiques quotidiennes.

Exemple :

$E = [[22.0, 27.4, 25.0, 64.8, 1, 1],$
 $[25.0, 31.2, 28.5, 14.2, 1, 1],$
 $[23.6, 32.2, 26.7, 0.0, 0, 0],$
 $[22.6, 34.0, 25.8, 0.0, 0, 0],$
 $[22.4, 34.2, 25.7, 23.1, 1, 1],$
 $[24.2, 28.3, 26.0, 14.3, 0, 0],$
 $[24.0, 28.8, 25.8, 12.9, 0, 0],$
 $...$
 $[24.2, 31.4, 27.1, 10.3, 1, 1]]$

Pour mesurer la similarité entre deux observations météorologique quotidiennes, on propose d'utiliser la distance euclidienne.

La distance euclidienne entre deux observations météorologiques quotidiennes X et Y est définie par la formule suivante :

$$\text{dist}(X, Y) = \sqrt{\sum_{i=0}^4 (X_i - Y_i)^2}$$

Q.14- Écrire la fonction $\text{dist}(X, Y)$ qui retourne la distance euclidienne entre deux observations météorologiques quotidiennes X et Y .

Exemple :

$X = [22.0, 27.4, 25.0, 64.8, 1, 1]$

$Y = [25.0, 31.2, 28.5, 14.2, 1, 1]$

$\text{dist}(X, Y)$ retourne le nombre 50.95144747698537

Q.15- Écrire la fonction $\text{tri}(E, R)$ qui reçoit en paramètres la liste E , et une liste R qui représente une observation météorologique un jour dans une station météorologique. La fonction trie les éléments de E dans l'ordre croissant de la distance euclidienne entre R et chaque élément de E . (Ne pas utiliser la fonction $\text{sorted}()$ et la méthode $\text{sort}()$)

Exemples :

$R = [23.8, 30.0, 26.1, 32.7, 1, 1]$

Après l'appel de la fonction $\text{tri}(E, R)$, on obtient la liste suivante :

$E = [[22.6, 32.2, 25.8, 30.6, 1, 1],$
 $[23.5, 32.2, 26.9, 30.0, 1, 1],$
 $[22.0, 31.8, 25.1, 30.2, 1, 1],$
 $[22.0, 31.8, 25.1, 30.2, 1, 1],$
 $\dots$
 $[24.0, 34.8, 28.2, 0.0, 0, 0],$
 $[23.8, 35.2, 28.3, 0.0, 0, 0],$
 $[22.0, 34.9, 28.4, 0.0, 0, 0],$
 $\dots$
 $[25.0, 30.6, 27.4, 89.0, 1, 1]]$

$[1.2, -2.2, 0.3, 2.1, 0, 0]$

$[(1.2)^2 + (-2.2)^2 + (0.3)^2 + (2.1)^2]$

$[-0.2, (-4.8), (-2.1), 32.7, 1, 1]$

$(0.2)^2 + (-4.8)^2 + (-2.1)^2 + (32.7)^2 + 1^2 + 1^2$

Q.16- Écrire la fonction de $\text{plus_proches}(E, R, k)$ qui reçoit en paramètres la liste E , une liste R qui représente une observation météorologique un jour dans une station météorologique, et k un entier strictement positif. La fonction retourne une liste contenant l'état de pluie le jour suivant des k observations météorologiques de E les plus proches à R .

Exemple :

$R = [23.8, 30.0, 26.1, 32.7, 1, 1]$

- $\text{plus_proches}(E, R, 9)$ retourne la liste $[1, 1, 1, 1, 1, 1, 1, 1, 0]$
- $\text{plus_proches}(E, R, 12)$ retourne la liste $[1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1]$

Q.17- Écrire la fonction de $\text{knn}(E, R, k)$ qui reçoit en paramètres la liste E , une liste R qui représente une observation météorologique un jour dans une station météorologique, et k un entier strictement positif. La fonction retourne l'état de pluie le jour suivant, le plus répété dans les états de pluie le jour suivant des k observations météorologiques de E les plus proches à R .

Exemple :

$R = [23.8, 30.0, 26.1, 32.7, 1, 1]$

- $knn(E, R, 9)$ retourne 1
- $knn(E, R, 11)$ retourne 1

B- Phase du test

Dans la phase du test du modèle, on utilise une liste T qui contient les 70% restant des observations météorologiques quotidiennes.

$T = [[22.0, 31.8, 25.1, 30.2, 1, 1],$
[26.2, 31.8, 28.5, 0.0, 0, 0],
[23.8, 31.3, 28.0, 8.5, 0, 0],
[23.6, 32.4, 27.5, 16.5, 1, 1],
[24.0, 30.2, 26.3, 3.4, 0, 0],
[25.4, 30.6, 27.6, 67.8, 1, 1],
[24.2, 31.5, 28.5, 1.2, 0, 0],
[24.6, 29.0, 26.0, 0.1, 0, 0],
...]

Q.18 - Écrire la fonction **test** (E, T, k) qui reçoit en paramètres les listes E et T , et un entier k strictement positif. En utilisant l'algorithme **KNN**, la fonction retourne une liste R telle que :

- Chaque élément R_i est un tuple à deux éléments ;
- Le premier élément du tuple est l'état de pluie réel le jour suivant, dans l'observation météorologique T_i ;
- Le deuxième élément du tuple est le l'état de pluie le jour suivant, prédit par l'algorithme KNN, pour l'observation météorologique T_i ;

Exemple :

La fonction **test** ($E, T, 13$) retourne la liste suivante :

$[(1, 1), (1, 1), (1, 1), (1, 1), (0, 0), (1, 1), (1, 1), (1, 1), (1, 1), (0, 0), (0, 0), (1, 1), (0, 0), (0, 1), (1, 1), (0, 0), (1, 1), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (1, 1), (0, 0), (0, 0), (0, 0), (1, 1), (1, 1), (0, 0), (0, 0), (1, 1), (1, 1), (0, 1), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (0, 0), (1, 1), (0, 0), (0, 0), \dots]$

Matrice de confusion

La matrice de confusion est un outil essentiel en apprentissage automatique pour évaluer la performance des modèles de classification en comparant les valeurs prédites aux valeurs réelles.

La matrice de confusion est une représentation tabulaire qui permet de visualiser les performances d'un modèle de classification. Elle compare les résultats réels (valeurs cibles) aux résultats prédits par le modèle. Chaque ligne de la matrice représente une classe réelle, tandis que chaque colonne représente une classe prédite. Cela permet d'identifier les prédictions correctes et incorrectes, ainsi que le type d'erreurs commises.

Structure de la matrice de confusion :

Dans sa forme la plus simple, la matrice de confusion est une matrice 2×2 pour les problèmes de classification binaire, qui contient les quatre catégories suivantes :

Vrais Positifs (VP)	Faux Négatifs (FN)
Faux Positifs (FP)	Vrais Négatifs (VN)

- ✓ **VP** : Prédictions positives correctes. Le vrai état de pluie le jour suivant est **1**, et l'état de pluie le jour suivant prédit par KNN est **1** ;
- ✓ **FP** : Prédictions positives incorrectes. Le vrai état de pluie le jour suivant est **0**, et l'état de pluie le jour suivant prédit par KNN est **1** ;
- ✓ **FN** : Prédictions négatives incorrectes. Le vrai état de pluie le jour suivant est **1**, et l'état de pluie le jour suivant prédit par KNN est **0**.
- ✓ **VN** : Prédictions négatives correctes. Le vrai état de pluie le jour suivant est **0**, et l'état de pluie le jour suivant prédit par KNN est **0** ;

Q.19 - Écrire la fonction **matrice_confusion** (*E*, *T*, *k*) qui reçoit en paramètres les listes *E* et *T*, et un entier strictement positif *k*. La fonction retourne la liste qui représente la matrice de confusion.

Exemple :

La fonction **matrice_confusion** (*E*, *T*, 13) retourne la matrice de confusion : **[[154, 2], [41, 450]]**

Métriques utilisées pour évaluer le modèle :

À partir des valeurs de la matrice de confusion, plusieurs métriques peuvent être calculées pour évaluer la performance du modèle :

- **Précision** : Mesure la proportion de prédictions positives correctes par rapport à toutes les prédictions positives.

$$\text{Précision} = \frac{VP}{VP + FP}$$

- **Rappel** : Mesure la capacité du modèle à identifier toutes les instances positives.

$$\text{Rappel} = \frac{VP}{VP + FN}$$

- **Exactitude** : Mesure la proportion de prédictions correctes (positives et négatives) par rapport à l'ensemble des prédictions.

$$\text{Exactitude} = \frac{VP + VN}{VP + FP + VN + FN}$$

- **Score F1** : La moyenne harmonique de la précision et du rappel, utile lorsque les classes sont déséquilibrées.

$$\text{Score F1} = 2 * \frac{\text{Précision} * \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

Q.20- Écrire la fonction **metriques** (*M*) qui reçoit en paramètre la matrice de confusion *M*, et qui retourne une liste contenant les valeurs, en pourcentage, de la **Précision**, du **Rappel**, de l'**Exactitude** et du **Score F1**.

Exemple :

La fonction **metriques** (*M*) retourne la liste **[88.33, 98.15, 96.29, 92.98]**

Partie III :

La stéganographie

L'art de dissimuler des données

Dans cette partie, on suppose que le module **numpy** est importé :

```
import numpy as np
```

La **stéganographie**, c'est l'art de dissimuler des données dans d'autres données. Il existe plusieurs techniques différentes qui permettent ce "tour de magie".

La stéganographie existe depuis longtemps, bien avant l'invention de l'ordinateur. Avec l'avènement de l'ordinateur et de son règne du "tout numérique", des techniques existent permettant pour cacher n'importe quel document dans un autre document. Les documents "porteurs" sont généralement des images (BMP, JPEG, ...) ou des sons (WAV, ...)

La stéganographie est utilisée dans l'industrie actuelle. Sa principale application est le watermarking. Cela consiste à cacher un copyright au sein d'une œuvre protégée. Ainsi, en cas de litige de droits d'auteurs, le watermark sera montré pour prouver l'originalité de l'œuvre.

Cette technique de dissimulation est aussi utilisée dans le monde des malwares pour infecter les ordinateurs grâce à des fichiers, à priori, innocents. Par exemple, le virus récupère une image dans laquelle se trouve un fichier qui contient les composants nécessaires à l'infection de l'ordinateur.

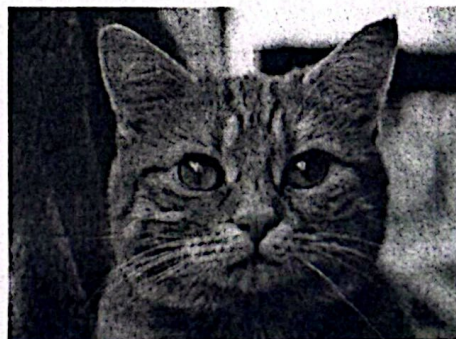
Dans la suite de cette partie, nous allons dissimuler une image en niveau de gris, dans une image couleur.

Une image numérique en niveaux de gris est représentée par une matrice à deux dimensions. Chaque élément de la matrice est appelé : **pixel**. Chaque pixel est représenté par un octet qui contient un nombre entier compris entre 0 et 255. Le nombre 0 correspond au noir, le nombre 255 correspond au blanc, et chacun des autres nombres 1, 2, 3, ..., 253, 254 correspond à un niveau de gris entre le noir et le blanc. Ainsi l'image est codée sur 256 niveaux de gris.

Exemple :

Matrice qui représente une image en niveau de gris.

```
[[26 35 32 ... 189 189 189]
 [29 31 33 ... 188 188 189]
 [32 32 33 ... 188 188 188]
 ...
 [30 31 31 ... 185 186 187]
 [32 32 32 ... 186 186 186]
 [34 33 34 ... 187 187 187]]
```



Q.21- Écrire la fonction **binaire(x)** qui reçoit en paramètre un entier **x** qui représente un pixel dans une image en niveaux de gris. La fonction retourne la liste de longueur **8**, qui représente la conversion binaire de **x**. Le bit du poids le plus faible se trouve à la fin de la liste.

avec justification.

paramètre une matrice M qui représente une image en et la conversion binaire de tous les pixels de M .

```
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1,
1, 0, 0, ... ]
```

être une liste contenant la conversion binaire d'un
urne la valeur décimale de ce pixel.

- e 26
e 189

er x , consiste à remplacer le dernier bit (de poids le
it de calculer la nouvelle valeur décimale de x .

0 par 1 : [0, 0, 0, 1, 1, 0, 1, 1]

12-20-2008 08:00 AM

un paramètre entier `x` qui représente un pixel dans la fonction remplace le bit de poids le plus faible dans la fonction retourne la nouvelle valeur décimale de `x`.

octets : le premier octet pour la composante **rouge**,
le deuxième octet pour la composante **bleue**. À partir de ces
deux octets, ce qui est largement plus que ne peut distinguer



```
[[ [73 121 45] [ 21 73 0] [25 80 0] [84 145 42] ... ]
 [21 73 0] [ 34 87 0] [66 123 26] [97 155 53] ... ]
 [30 88 0] [ 77 135 33] [97 158 54] [67 128 24] ... ]
 [65 129 15] [100 164 52] [91 155 45] [34 97 0] ... ]
 [71 140 21] [ 79 146 31] [65 130 26] [39 98 6] ... ]
...
]
```

Principe pour dissimuler une image en niveaux de gris dans une image en couleurs :

L'astuce est d'utiliser le bit de poids le plus faible dans chaque octet de chaque pixel de l'image couleur. En modifiant ce bit, on modifie un peu l'image, mais ce n'est pas visible à l'œil nu :

- On converti l'image en niveaux de gris en une liste de bits B ;
- On dissimule successivement chaque bit de la liste B dans chaque octet de chaque pixel de l'image couleur.

Exemple :

$$B = [0, \quad 0, \quad 0, \quad 1, \quad 1, \quad 0, \dots]$$

$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow$

$[73 \quad 121 \quad 45] [21 \quad 73 \quad 0] \dots$

Pixel Pixel

- On dissimule le bit 0 dans le nombre 73
- On dissimule le bit 1 dans le nombre 121
- On dissimule le bit 0 dans le nombre 45
- ...

Q.26- Selon le principe ci-dessus, quelle est la taille maximale en **MO** de l'image en niveaux de gris, qu'on peut dissimuler dans une image en couleurs de taille **1GO** ?

Q.27- Écrire la fonction **dissimule_img (C, M)** qui reçoit en paramètres une matrice **C** qui représente une image en couleurs, et une matrice **M** qui représente une image en niveaux de gris. La fonction dissimule l'image **M** dans l'image **C**, en utilisant le principe ci-dessus. (On suppose que la taille de **C** est suffisante pour la dissimulation de **M**)

Exemple :

