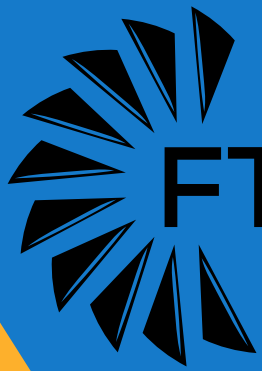
PROBLÈMES DE PROGRAMMATION
SÉNIOR DE ROBOTIQUE CRC

PRÉLIMINAIRE 4

MODUEL 2026

présenté par



FTAI AVIATION

Un programme de

AEST EAST

Version 1.0

QUELQUES NOTES

- Les règles complètes sont dans la section 4 du livret des règlements.
- Vous avez jusqu'au **vendredi 13 février, 23h59** pour remettre votre code.
- N'hésitez pas à utiliser le forum de programmation sur le discord de la CRC pour poser vos questions et discuter des problèmes. Il est là pour ça!
- **On vous donne des fichiers modèles faciles à utiliser pour votre code et pour faire vos tests. Vous devez les utiliser pour résoudre le problème!**

UTILISATION DU FICHIER MODÈLE

- Le fichier de test appelle la fonction associée avec en paramètre les informations du test et compare sa sortie avec ce qui est attendu pour vous permettre de voir si les tests réussissent. **Tout votre code (sauf fonctions additionnelles que vous créez) devrait être écrit dans la fonction prévue à cet effet.**
- Les points mis dans le document indiquent la difficulté et le pointage attribué pour la réussite pour chaque défi. Ce problème préliminaire aura une valeur globale de 2% du défi principal.

STRUCTURE

Une petite mise en situation comme celle-ci explique les fondements de chaque défi et offre les bases nécessaires pour résoudre celui-ci.

Spécification d'entrée et de sortie:

Contient les caractéristiques des entrées fournies ainsi que les critères attendus pour les sorties du programme.

Exemple d'entrée et de sortie:

Contient un exemple d'entrée, parfois constitué lui-même de plusieurs sous-exemples, pour que vous puissiez tester votre programme. Chaque exemple de sortie donne la réponse attendue pour l'entrée correspondante.

Explication de la première sortie:

Décortique davantage le défi en expliquant comment la première entrée est traitée et en montrant le chemin menant à cette réponse.

Patrons et répétitions!

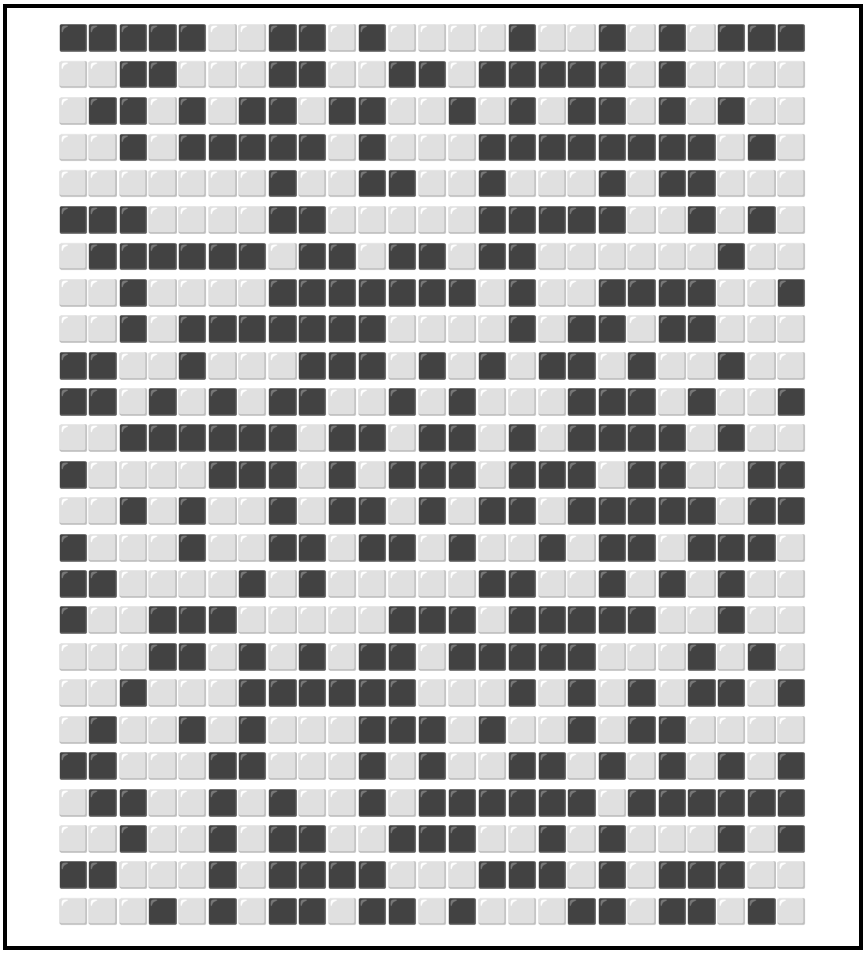
À mon âge avancé de… 24 ans, je, Frank, commence à me répéter souvent. Certains y voient un problème, mais moi j'y trouve des patrons et des symétries intéressantes. Plus on cherche, plus on trouve des répétitions et des patrons dans la vie de tous les jours, que ce soit dans des mots, des chiffres, des tuiles, des montagnes ou même des vagues.

Partie 1: Tuiles de bain (X points)

Vous venez tout juste de faire la rénovation de votre salle de bain et vous avez demandé à l'installateur de tuiles de les placer de manière aléatoire, mais vous ne pouvez vous empêcher de trouver les mêmes motifs qui se répètent…

À partir du plan des tuiles de votre salle de bain, vous devrez compter le nombre de fois qu'un motif donné apparaît.

Voici le plan du plancher :



Spécification d’entrée et de sortie:

Vous recevrez une *list* de *list* (tableau 2D) qui représente le motif à trouver dans la grille de la salle de bain. Celle-ci sera intégrée dans votre fichier Python. Attention! Les 0 dans les motifs indiquent qu’une tuile noire ***OU*** blanche peut se situer à cet emplacement.

Vous devez retourner un *int* qui représente le nombre de fois où le motif donné apparaît. Il est important de noter que les motifs ne doivent pas se chevaucher durant votre compte. C’est à vous de trouver l’arrangement optimal du motif pour en rentrer le plus grand nombre possible.

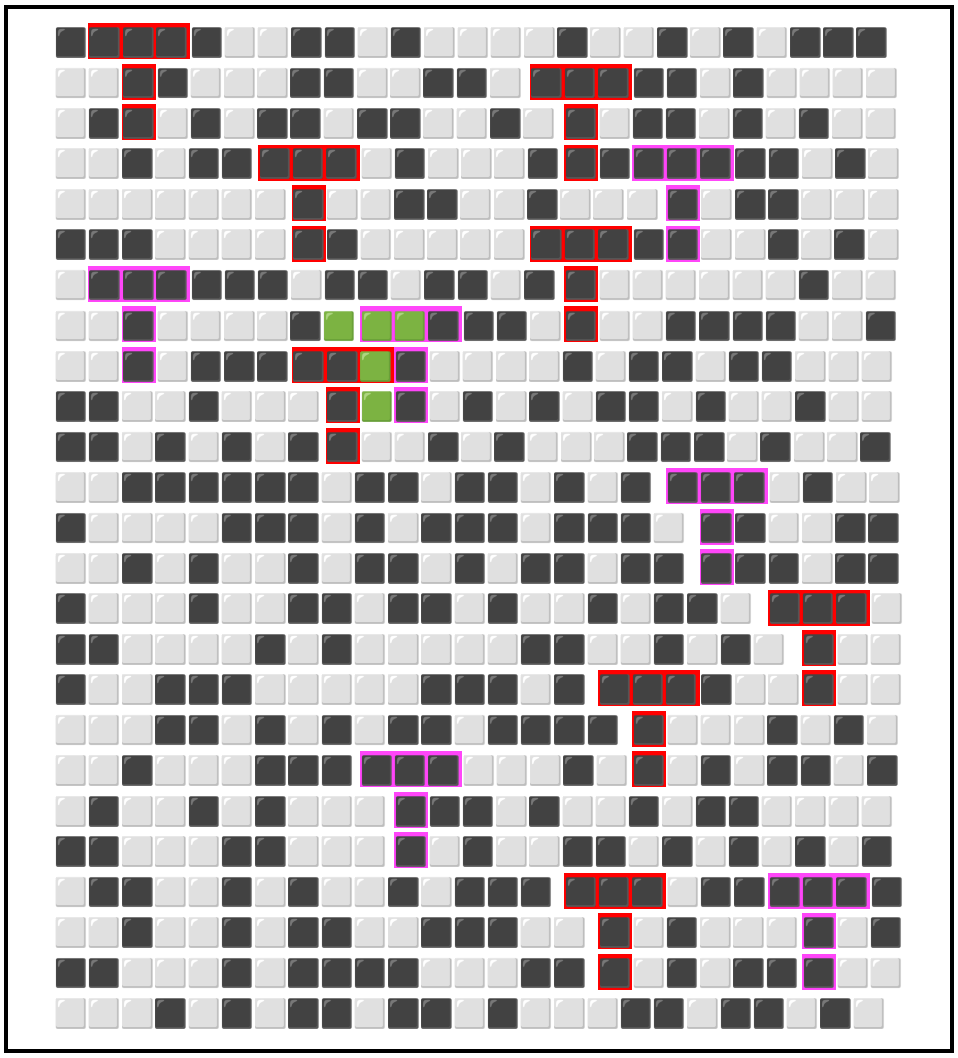
Exemple d’entrée:

```
[ [1, 1, 1],
  [0, 1, 0],
  [0, 1, 0] ]

[ [0, 0, 1],
  [0, 1, 1],
  [1, 1, 0] ]

[ [1, 1],
  [1, 1] ]

[ [1, 0],
  [1, 0],
  [1, 1],
  [0, 1] ]

[ [1, 1],
  [0, 1],
  [1, 1] ]
```

Exemple de sortie:

```
14
14
25
13
12
```

Explication de la première sortie:



Même si le motif peut être repéré à d'autres endroits, comme le *«T»* en vert, seulement 14 peuvent être placés sur le plan en même temps.

Partie 2: Palindromes (X points)

Les palindromes sont des mots, des nombres ou des séquences de caractères qui sont pareillement lus normalement ou en partant de la fin. Par exemple, le mot *«Laval»* ou la date *«12/02/2021»* sont des palindromes.

Mais pour ce problème, on s'intéresse aux nombres qui sont aussi des palindromes. Vous devrez résoudre les équations donnés et déterminer si l'entier résultant est un palindrome. Comme nous souhaitons mettre l'emphase sur les palindromes et non la résolution d'équation, vous n'avez pas besoin de vous soucier de la priorité des opérations selon les règles conventionnelles. Appliquer les opérations de gauche à droite sera suffisant pour ce problème.

Spécification d'entrée et de sortie:

En entrée, vous recevrez une équation mathématique sous la forme d'une *string* de nombres et d'opérations arithmétiques de base (+, -, x, ÷).

Vous devez retourner *Yes* si le résultat de l'équation est un palindrome ou *No* si ce n'en est pas un.

Exemples d'entrée:

36363×11

121×343+55255

572×471+550

254168116÷4-62999784

Exemples de sortie:

```
Yes

No

Yes

Yes
```

Explication de la première sortie:

La première équation donne 399993 et à l'envers le nombre reste 399993, donc c'est un palindrome.

Partie 3: Mosaïque parfaite (X points)

Vous faites un plancher en mosaïque pour lequel vous utilisez des tuiles carrées. Ce plancher est composé de tuiles 1x1 soit noire (@) ou blanche (#) sous forme de grille. Voici un exemple d'une tuile complète utilisant les plus petites tuile 1x1:

@#
#@

Nous avons dans cet exemple une tuile 2x2 contenant des tuiles noires aux coins en bas à droite ainsi qu'en haut à gauche avec des tuiles blanches dans les deux autres coins. Voici un exemple de tuile 3x3:

@@#
@##
###

Votre tâche est de trouver le moyen de faire le plancher de M par N avec des tuiles carrées pour obtenir le coût le plus faible possible. **Rappelez-vous que les tuiles utilisées doivent être carrées et toutes de la même grandeur**. Chaque design de tuile que vous choisissez dans la mosaïque peut être utilisé comme la même sorte de tuile même si on y applique une rotation de 90, 180 ou 270 degrés.

Nous vous fournirons les dimensions du plancher de mosaïque avec M étant la hauteur et N étant la largeur du plancher. Vous aurez aussi une représentation du plancher pour lequel vous devrez déterminer le découpage des tuiles pour obtenir un prix minimal.

Le coût pour la fabrication du plancher peut être calculé à l'aide de la formule suivante:

$$(50 + floor(\frac{360}{s}))i + 80t$$

Voici les différentes portions de la formule:

- Nous avons tout d'abord $(50 + floor(\frac{360}{s}))i$ qui représente le coût de créer un patron de tuile unique. Chaque patron de tuile unique a un coût de design de 50$ auquel la dimension de la tuile s'ajoute. Chaque tuile est moins chère plus elle est grande et coûte 360$/s avec s étant la longueur d'un côté de la tuile qu'on design. Une tuile 2x2 aurait s=2 alors qu'une tuile 3x3 a un s=3. Le i par lequel est multiplié la première partie de la formule est le nombre de tuiles uniques qui sont créées.

- La seconde partie du prix est le nombre de tuiles ayant besoin d'être mises pour former le plancher et indépendamment de sa dimension, chaque tuile coûte 80$ à poser.

Petit rappel, la fonction floor(x) nous donne la valeur entière plus faible. Donc la valeur floor(3.7) = 3

Vous devrez donc trouver le coût le plus faible (en faisant un découpage en tuiles carrées) pour faire le plancher de mosaïque qui vous sera fourni.

Spécification d'entrée et de sortie:

En entrée, vous recevrez deux *integer* M et N qui sont la hauteur ainsi que la largeur du plancher de mosaïque. Vous recevrez aussi un *array* de *string* qui sont la représentation du plancher mosaïque. Ces *string* seront composés des tuiles individuelles blanches (#) et noires(@) que vous pourrez (ou non) découper en tuiles pour faire le plancher le moins coûteux possible.

En sortie vous devrez donner un *integer* du prix minimal pour la mise en place du plancher en mosaïque.

Exemple d'entrée:

```
M=3 N=3
@@#
@##
###

M=2 N=4
@@##
##@@

M=4 N=8
@@@@@#@#
@@@@#@#@
@@@@@#@#
@@@@#@#@
```

Exemple de sortie:

```
250

390

440
```

Explication de la première sortie:

Dans le premier exemple, nous avons 2 options, soit nous formons une tuile 3x3 ou nous avons 9 tuiles 1x1.
Si nous faisons 9 tuiles 1x1 avec 2 variantes (une tuile unique noire et une tuile unique blanche) Voici le coût de production:

$$(50 + 360) \times 2 + 80 \times 9 = 1540$$

Pour la seconde option qui est de faire une grande tuile de grandeur 3x3 avec le coût suivant:

$$(50 + 360/3) + 80 * 1 = 250$$

Nous prenons donc l'option d'une tuile 3x3 qui est moins chère que 9 tuiles 1x1 avec 2 designs.

Explication de la deuxième sortie:

Nous avons 2 options, soit une tuile 2x2 installée 2 fois ou plutôt 2 sortes de tuiles 1x1 installées 8 fois. Regardons tout d'abord le coût des 8 tuiles individuelles.

$$(50 + 360) \times 2 + 80 \times 8 = 1460$$

Explorons maintenant l'option d'utiliser 2 tuiles avec le design suivant:
@@
##
Notez que le design est équivalent à sa rotation pour compléter le plancher de mosaïque et ainsi une seule sorte de tuile a besoin d'être design. Nous avons le coût suivant:

$$(50 + 360/2) + 80 \times 2 = 390$$

Nous pouvons donc conclure qu'il est moins coûteux de faire une tuile unique 2x2 et la poser à 2 reprises pour un coût total du plancher de 390$.

Partie 4: Convexe ou concave? (20 points)

En utilisant une liste de points 2D qui vous sera donnée, vous devrez déterminer s'il est possible de créer un polygone convexe en reliant les points donnés.

ps.: Il est possible de transformer les coordonnées cartésiennes en coordonnées polaires. C'est à vous de décider si cela peut vous être utile ou non. Vous pouvez aussi utiliser des logiciels de dessin de plan mathématique comme GeoGebra et Desmos pour vous aider à visualiser les listes de points données pour penser à comment attaquer le problème.

Spécification d'entrée et de sortie:

En entrée, vous allez recevoir une liste de coordonnées cartésiennes (x, y).

En sortie, vous devrez donner un *bool* qui représente s'il est possible de relier les points pour créer un polygone convexe.

Exemple d'entrée:

[(4, 0), (0, 4), (-4, 0), (0, -4)]
[(5, 1), (6, 4), (2, 8), (-1, 6), (-6, 8), (-8, 3)]
[(-3, 5), (0, 0), (10, 0), (-7, 5), (3, 5), (-10, 0), (0, -10), (7, 5)]
[(-6, -15), (3, 3), (-2, 3), (4, 18), (-8, 0), (-4, 0)]

Exemple de sortie:

```
True
False
False
False
```

Explication de la première sortie:

Servez-vous de dessins et d'outils graphiques pour vous trouver une stratégie de résolution. Plusieurs méthodes pour résoudre sont possible, utilisez votre créativité et développez votre sens de la résolution de problème.