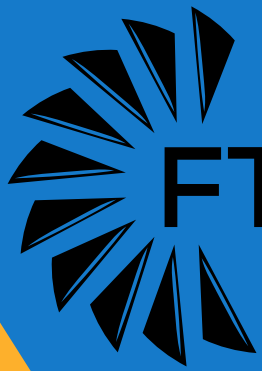
PROBLÈMES DE PROGRAMMATION
SÉNIOR DE ROBOTIQUE CRC

PRÉLIMINAIRE 2

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 5 décembre, 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.

Jeux de société!

Que ce soit lors d'une fin de semaine pluvieuse, un soir de semaine ou même à n'importe quel moment de la journée, les jeux de société créent des moments inoubliables. Nombreuses sont les amitiés qui se sont développées autour d'un bon jeux de société et aujourd'hui nous allons explorer certains des jeux de société les plus importants dans la vie des organisateurs de CRC.

Partie 1: Serpent échelles (20 points)

Serpent échelles est un jeu sur une grille de 100 cases avec des serpents qui font descendre les pions et les échelles les font monter. En roulant un dé traditionnellement à 6 faces, on déplace les pions de la quantité de case indiqué sur le dé.

Vous allez devoir trouver le nombre minimal de lancé de dé, vous pouvez rouler le nombre que vous voulez sur le dé. Vous commencez hors du plateau, donc si votre premier lancer est 6 vous allez sur la case 6.

Spécification d'entrée et de sortie:

Vous recevrez un *int* pour le nombre de face sur le dé. Une liste de *tuple* pour les échelles et une autre pour les serpents.

Vous devez retourner un *int* du nombre de lancé minimal pour atteindre la case 100.

Exemple d’entrée:

d = 6
Échelles : {(1, 38), (4, 14),(21, 42), (28, 84)}
Serpents : {(48, 26), (49, 11), (62, 19), (87, 24)}

d = 8
Échelles : {(7, 50)}
Serpents : {(80, 30)}

d = 10
Échelles : {(4, 9), (41, 99)}
Serpents : {(31, 5), (50, 20)}

Exemple de sortie:

7
8
6

Explication de la première sortie:

Avec un dé à 6 faces, le plus qu'on peut bouger est de 6 cases. L'échelle la plus longue est (28, 84), pour se rendre à cette échelle, il faut ignorer la première échelle et se rendre à l'échelle (4, 14) en un lancer. Ensuite, on ne peut aussi pas utiliser l'échelle (21, 42) et il faut lancer pour aller à 28 avec 3 lancers. Après avoir monté l'échelle, on doit rouler 3 fois pour se rendre à la fin, ce qui nous donne 7 lancers.

Partie 2: Scrabble (40 points)

Le jeu de scrabble consiste à placer des lettres sur un plateau dans le but de former des mots et faire des points. Ces mots peuvent être placés de façon orthogonale soit vers le haut, le bas, à droite ou à gauche. Une lettre posée sur le plateau peut être partagée par deux mots, un qui est horizontal et un qui est vertical.

Le but du problème va être de placer un mot en le faisant **croiser le plus de mot** déjà présent sur le plateau de jeu.

Spécification d'entrée et de sortie:

En entrée, vous recevez une variable de type *liste* contenant des *listes* de *string* forment les mots déjà placés et une variable de type *string* contenant un mot que vous devrez placer dans le terrain de scrabble afin de faire le plus de point possible.
En sortie, vous devrez donner une *liste* formée de *liste* de *string* qui est le plateau de scrabble après avoir placé le mot.

Exemple d'entrée:

```
[[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', 'T', ' ', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', 'O', ' ', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', 'U', ' ', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', 'J', ' ', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', 'O', ' ', ' ', 'D', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', 'U', ' ', ' ', 'N', ' ', ' ', ' '],
 [' ', ' ', 'S', 'E', 'R', 'T', 'T', 'E', 'L', ' ', ' '],
 [' ', ' ', ' ', ' ', 'S', ' ', ' ', 'I', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', ' ', ' ', ' ', 'R', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', ' ', ' ', ' ', 'F', ' ', ' ', ' ']]
```

“BOLIDES”

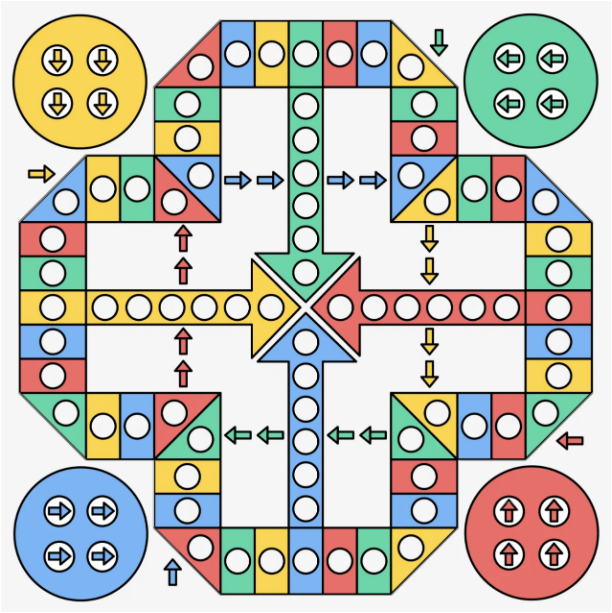
Exemple de sortie:

```
[[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', 'T', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', 'O', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', 'U', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', 'J', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', 'B', 'O', 'L', 'I', 'D', 'E', 'S', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', 'U', ' ', ' ', 'N', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', 'S', 'E', 'R', 'T', 'T', 'E', 'L', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', 'S', ' ', ' ', 'I', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', ' ', ' ', ' ', 'R', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', ' ', ' ', ' ', 'F', ' ', ' ', ' ', ' ', ' ']]
```

Explication de la première sortie:

Le mot “BOLIDE” est placé de manière à partager la lettre O avec le mot “TOUJOURS” et partage la lettre D avec le mot “FRIEND”. C’est le seul emplacement à considérer comme c’est celui qui permet de croiser le plus de mots possibles.

Partie 3: Aeroplane Chess (85 points)



Le jeu d'aéroplane est un jeu de société populaire, joué avec 2 à 4 joueurs. Chaque joueur (appelé une « équipe » dans ce document) possède une couleur (Rouge, Bleu, Jaune ou Vert), qui sert à identifier l'équipe par la suite, et 4 avions. L'objectif du jeu est d'être la première équipe à faire voler ses 4 avions depuis la zone de départ («hangar») située dans un coin du plateau jusqu'à la destination (le centre du plateau).

Une image du plateau du jeu d'aéroplane *original* se trouve à gauche.

En plus d'«équipe», «avion», et «hangar», voici d'autres termes importants à connaître :

- Le «chemin principal» est le chemin circulaire autour du plateau que tous les avions des équipes doivent parcourir. Il comporte 52 cases, et tous les avions doivent suivre ce chemin dans le sens ***horaire***.
- Le «chemin final» est le chemin coloré qui mène du chemin principal à la destination (centre) pour chaque équipe. Chaque chemin final comporte 6 cases et a la forme d'une grande flèche pointant vers le centre. Une fois que l'avion arrive sur la case du chemin principal juste avant le chemin final, il DOIT entrer sur le chemin final à la marche suivante. L'avion n'a PAS le droit d'avancer plus de pas que nécessaire pour atteindre la destination (c'est-à-dire qu'il ne peut pas dépasser la case d'arrivée).
- * La «zone de lancement» d'une équipe est la case du chemin principal avec une flèche de la même couleur que l'équipe pointant vers la case. C'est là que l'avion entre sur le chemin principal depuis le hangar.
    - Pour éviter la confusion, sur l'image, les zones de lancement sont toutes des formes triangulaires et **ne sont pas** colorées selon les couleurs des équipes.
- L'«aérodrome» (airway) est la paire de cases ***colorées*** sur le chemin principal avec des flèches directionnelles (de la même couleur) les reliant. Un avion qui s'arrête sur la case avec une flèche pointant vers l'autre case est immédiatement déplacé vers la case associée, et cela *ne compte pas* comme un tour.

➢ Un «tour» est le processus par lequel un avion se déplace d'une case à une autre, ce qui peut impliquer l'utilisation de gadgets.

Vous jouez maintenant une version spéciale de ce jeu **avec un seul avion par équipe**. Vous recevrez une liste contenant 52 strings représentant toutes les cases du chemin principal, en commençant depuis la fin du chemin final de l'équipe Verte et en avançant dans le sens horaire. Chaque string aura le format suivant :

`[L<COLOR>]<COLOR>#<STEP>#[G<F/C/T>][<E/T>][F]`

où :

- ***[L<COLOR>]*** peut être présent ou non. S'il est présent, il indique que cette case est la zone de lancement de l'aérodrome pour l'équipe de couleur correspondante. Par exemple : ***[LR]*** indique que cette case est la zone de lancement de l'équipe Rouge sur le chemin principal, et de même pour ***[LB]***, ***[LY]***, et ***[LG]***.
- ***<COLOR>*** est l'une des lettres 'R', 'B', 'Y' ou 'G', représentant respectivement Rouge, Bleu, Jaune ou Vert.
- ***<STEP>*** est un entier entre 1 et 6 (inclus), représentant le nombre de pas que l'avion sur cette case doit obligatoirement avancer au prochain tour. Un peu comme un roulement de dé. Toutefois, si l'avion se trouve sur une case de la même couleur que lui, il *peut choisir* d'avancer n'importe quel nombre de pas entre 1 et ***<STEP>*** (inclus).
- ***[G<F/C/T>]*** est un code à deux caractères qui peut être présent ou non. S'il est présent, il indique un gadget pouvant affecter l'avion sur cette case.
    - ***[GF]*** est un «gadget : ventilateur» qui permet à l'avion d'avancer 3 pas supplémentaires à son prochain tour (si l'avion n'est pas sur une case de même couleur), ou jusqu'à 3 pas supplémentaires (s'il est sur une case de même couleur). Par exemple, si vous êtes sur une case `B#6#[GF]` et que vous choisissez d'utiliser le gadget : si vous êtes rouge, vous devez avancer de 9 cases, mais si vous êtes bleu, vous pouvez avancer de 1 à 9 cases au tour suivant. La même logique s'applique pour chaque gadget.
    - ***[GC]*** est un «gadget : cœur» qui double le nombre de pas que l'avion peut avancer au prochain tour (s'il n'est pas sur une case de même couleur), ou double le nombre maximum de pas possibles (si l'avion est sur une case de même couleur).
    - ***[GT]*** est un «gadget : turbine» qui a le même effet que ***[GT]***, mais au lieu de doubler, il triple le nombre de pas (ou le maximum possible).

    - L'équipe peut choisir de ne pas utiliser le gadget, mais tout gadget non utilisé est perdu et n'est pas stocké pour les tours futurs.
- ➢ ***[E/T]*** peut être présent ou non. S'il est présent, il indique que cette case fait partie d'un «aérodrome» pour l'équipe de couleur ***<COLOR>***. 'E' indique la case d'entrée de l'aérodrome, tandis que 'T' indique la case cible. Si un avion de la couleur correspondante atterrit sur la case ***[E]***, il est immédiatement déplacé vers la case ***[T]***.
- ➢ ***[F]*** peut être présent ou non. S'il est présent, il indique que cette case est le début du chemin final pour l'équipe de couleur ***<COLOR>***.

Un exemple de string valide est :

[LB]R#4#[GC][T]

Ce qui indique que cette case :

- ➢ est la zone de lancement de l'aérodrome de l'équipe Bleue,
- ➢ est une case de couleur Rouge,
- ➢ permet à un avion dessus d'avancer (jusqu'à) 4 pas,
- ➢ possède un gadget «cœur» qui double les pas possibles au prochain tour,
- ➢ est une case cible de l'aérodrome pour l'équipe Rouge.

Vous recevrez également quatre listes de 6 strings chacune représentant les cases du chemin final pour chaque équipe, dans l'ordre Rouge, Bleu, Jaune, Vert. Chaque chaîne aura le même format que pour le chemin principal sauf qu'il n'y a pas de ***[<E/T>][F]*** à la fin puisque les cases du chemin final ne font pas partie d'un aérodrome ni du début d'un chemin final.

On vous donnera également une couleur représentant votre équipe (c'est-à-dire votre couleur). Pour maximiser vos chances de gagner, vous devrez explorer le plus court chemin possible pour que votre avion atteigne la destination, en tenant compte de tous les gadgets et cases spéciales du plateau. En sortie, indiquez le nombre minimal de tours nécessaires pour que votre avion atteigne la destination.

Notes:

- ➢ La couleur des cases suit strictement la disposition originale du plateau d'aéroplane (c'est-à-dire commence par 'G', puis 'R', 'B', 'Y', et répète).
- ➢ La case d'aérodrome d'une équipe aura toujours son entrée à 17 cases après la zone de lancement correspondante, et sa cible à 29 cases après.
- ➢ La case au début du chemin final pour chaque équipe sera toujours la 49e case après la zone de lancement correspondante.
- ➢ Il n'y a qu'une seule zone de lancement, une seule case d'entrée d'aérodrome, une seule case cible d'aérodrome et une seule entrée de chemin final pour chaque équipe sur le chemin principal.
- ➢ Ces règles sont toujours respectées dans les données d'entrée, et elles suivent strictement la conception du plateau original du jeu d'aéroplane. Ces étiquettes sont également fournies dans les données d'entrée pour vous aider à déterminer les couleurs et positions de ces cases spéciales.

Spécification d'entrée et de sortie:

- ➔ Vous recevrez d'abord un tableau de 52 *string* représentant les cases du chemin principal, en commençant par la fin du chemin final de l'équipe Verte et en avançant dans le sens horaire.
- ➔ Ensuite, vous recevrez quatre *array* de 6 *string* chacun, représentant les cases du chemin final pour les équipes Rouge, Bleu, Jaune et Verte.
- ➔ Enfin, vous recevrez une *string* représentant votre couleur d'équipe ('R', 'B', 'Y' ou 'G').
- ➔ Vous devrez afficher un *int* représentant le nombre minimal de tours nécessaires pour que votre avion atteigne la destination.

Exemples d'entrée:

Exemple d'entrée 1

```
["G#4#[F]", "R#1#", "B#1#", "[LG]Y#4#", "G#1#", "R#4#", "B#6#[T]",
"Y#5#[E]", "G#2#", "R#2#", "B#2#", "Y#4#", "G#4#", "R#1#[GF][F]",
"B#2#", "Y#2#", "[LR]G#3#", "R#5#", "B#6#[GC]", "Y#2#[T]", "G#2#[E]",
"R#6#", "B#3#", "Y#4#", "G#2#", "R#5#", "B#2#[F]", "Y#4#", "G#4#",
"[LB]R#3#", "B#1#", "Y#3#", "G#4#[T]", "R#3#[E]", "B#1#", "Y#4#",
"G#4#[GF]", "R#3#", "B#2#", "Y#4#[F]", "G#2#[GC]", "R#6#",
"[LY]B#4#", "Y#1#", "G#4#", "R#6#[T]", "B#5#[GC][E]", "Y#4#", "G#3#",
"R#5#", "B#2#", "Y#6#"]
["#3#[GC]", "#4#", "#3#", "#6#", "#6#", "#5#"]
["#6#", "#3#", "#4#", "#5#", "#6#", "#1#"]
["#4#", "#5#", "#3#", "#4#", "#6#", "#6#"]
["#3#", "#4#", "#2#", "#2#", "#3#", "#2#"]
'B'
```

Exemple d'entrée 2

```
["G#6#[F]", "R#6#[GF]", "B#2#[GT]", "[LG]Y#6#", "G#5#[GF]",
"R#1#[GC]", "B#6#[GC][T]", "Y#3#[E]", "G#6#[GF]", "R#4#", "B#5#",
"Y#3#", "G#6#[GT]", "R#1#[F]", "B#2#", "Y#4#", "[LR]G#5#", "R#6#",
"B#6#", "Y#1#[T]", "G#4#[GF][E]", "R#2#", "B#2#", "Y#4#[GF]", "G#5#",
"R#4#", "B#5#[GC][F]", "Y#2#", "G#2#", "[LB]R#6#", "B#6#",
"Y#3#[GF]", "G#3#[T]", "R#3#[E]", "B#4#", "Y#1#", "G#6#", "R#1#",
"B#2#", "Y#3#[F]", "G#6#", "R#2#", "[LY]B#6#[GC]", "Y#4#", "G#6#",
"R#6#[T]", "B#2#[E]", "Y#4#[GF]", "G#4#", "R#2#", "B#4#", "Y#2#"]
["#3#", "#5#", "#2#[GC]", "#3#", "#4#[GF]", "#3#[GF]"]
["#3#[GC]", "#6#", "#1#", "#5#[GF]", "#6#", "#6#"]
["#4#", "#5#", "#4#[GF]", "#1#", "#1#", "#6#"]
["#6#", "#1#", "#5#[GF]", "#2#", "#4#", "#2#"]
'R'
```

Exemple d'entrée 3

```
["G#4#[F]", "R#2#", "B#4#", "[LG]Y#2#", "G#3#", "R#4#", "B#4#[T]",
"Y#6#[E]", "G#1#", "R#1#", "B#5#", "Y#6#", "G#1#[GF]", "R#6#[GF][F]",
"B#1#", "Y#2#", "[LR]G#4#", "R#3#", "B#2#", "Y#1#[T]", "G#3#[E]",
"R#1#[GC]", "B#2#", "Y#3#", "G#3#", "R#6#", "B#2#[F]", "Y#6#",
"G#6#", "[LB]R#6#", "B#4#", "Y#4#", "G#3#[T]", "R#4#[E]", "B#1#",
"Y#1#", "G#2#", "R#6#", "B#6#", "Y#2#[F]", "G#6#", "R#5#",
"[LY]B#1#", "Y#2#", "G#5#", "R#5#[T]", "B#3#[E]", "Y#3#", "G#1#",
"R#5#", "B#2#", "Y#3#"]
["#1#", "#1#", "#1#", "#2#", "#4#", "#2#[GT]"]
["#4#", "#1#", "#2#", "#2#", "#2#", "#1#"]
["#3#[GF]", "#2#[GF]", "#6#[GC]", "#2#[GF]", "#5#", "#3#"]
["#3#", "#6#", "#1#", "#1#", "#5#", "#3#"]
```

```
'Y'
```

Exemples de sortie:

Exemple de sortie 1
```
11
```

Exemple de sortie 2
```
9
```

Exemple de sortie 3
```
14
```

Explication de la première sortie:

L'équipe bleue prend au minimum **11 tours** pour arriver à destination.

Voici le chemin suivi:

- Tour 1: Sur le chemin principal position 0 -> Avance 3 pas vers la position 3
- Tour 2: Sur le chemin principal position 3 -> Avance 4 pas vers la position 7
- Tour 3: Sur le chemin principal position 7 -> Avance 4 pas vers la position 11
- Tour 4: Sur le chemin principal position 11 -> Avance 2 pas vers la position 13
- Tour 5: Sur le chemin principal position 13 -> Avance 4 pas vers la position 17
- **Tour 5: Automatiquement prend le "airway" vers la position 29**
- Tour 6: Sur le chemin principal position 29 -> Avance 6 pas vers la position 35
- Tour 7: Sur le chemin principal position 35 -> Avance 4 pas vers la position 39
- Tour 8: Sur le chemin principal position 39 -> Avance 3 pas vers la position 42
- Tour 9: Sur le chemin principal position 42 -> Avance 2 pas vers la position 44
- Tour 10: Sur le chemin principal position 44 -> Avance 6 pas vers le chemin final position 0
- Tour 11: Du chemin final position 0 -> Avance 5 pas jusqu'à la DESTINATION

Partie 4: Carcassonne (65 points)

Dans le jeu Carcassonne, basé sur la ville française du même nom, les joueurs piochent chacun leur tour une tuile qu'ils doivent placer sur le jeu pour agrandir la ville et faire des points. La pose des tuiles obéit toutefois à une règle essentielle: chaque côté d'une nouvelle tuile doit correspondre au type de terrain des tuiles adjacentes. Dans la version de base, on retrouve trois types de terrains: les prés, les routes et les châteaux.

Le but de votre programme va être de trouver une liste de positions valides pour placer une tuile sur un plateau de jeu donné. Pour ce faire, vous disposez de 2 classes pour gérer les tuiles et le plateau de jeu. Certaines fonctions vous sont déjà données et/ou recommandées, mais vous pouvez ajouter les vôtres au besoin.

Spécification d'entrée et de sortie:

En entrée, vous recevrez un objet de type **Tile** ainsi qu'un objet de type **Board** préremplie. Une tuile est représentée par une chaîne de quatre caractères, où chaque caractère décrit le type de terrain présent sur un côté de la tuile. L'ordre des caractères est toujours le même:

- le premier correspond au **côté supérieur**,
- le deuxième au **côté droit**,

- le troisième au **côté inférieur**,
- et le quatrième au **côté gauche**.

Vous pourrez accéder à ces valeurs avec les propriétés **.top**, **.right**, **.bottom** et **.left** . Ensuite, les types de terrains sont codés à l'aide de lettres:

- **g** pour les **prés**,
- **r** pour les **routes**,
- **c** pour les **châteaux**,
- **n** pour une **tuile blanche**.

Un **Board** est simplement une liste à deux dimensions contenant des objets **Tile**. Chaque case du plateau peut soit être vide (une tuile blanche), soit contenir une tuile avec de l'information.

Exemple d'entrée:

Exemple 1:

```
tuile = Tile("ggrr")
board = [["nnnn", "nnnn", "nnnn"],
         ["nnnn", "crgr", "nnnn"],
         ["nnnn", "nnnn", "nnnn"]]
```

Exemple 2:

```
tuile = Tile("rgrr")
board = [["nnnn", "nnnn", "nnnn", "nnnn"],
         ["nnnn", "crgr", "ccrr", "nnnn"],
         ["nnnn", "nnnn", "rgrr", "nnnn"],
         ["nnnn", "nnnn", "nnnn", "nnnn"]]
```

Exemple 3:

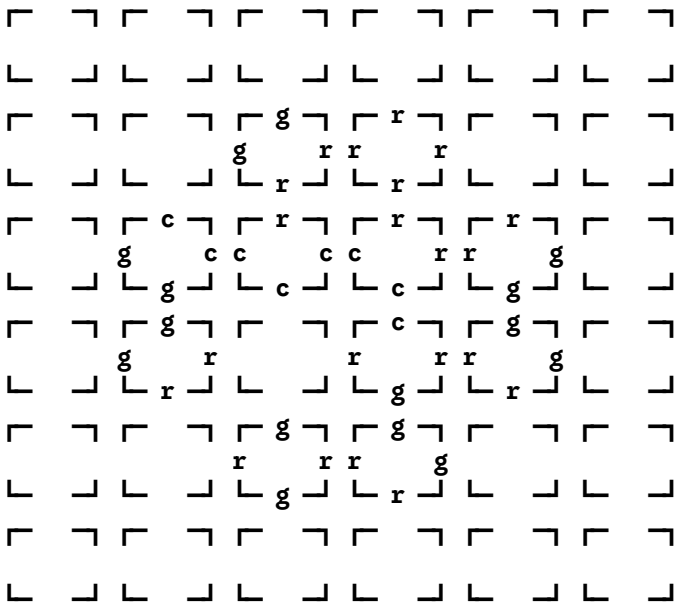
```
tuile = Tile("crgr")
board = [["nnnn", "nnnn", "nnnn", "nnnn", "nnnn", "nnnn"],
         ["nnnn", "nnnn", "grrg", "rrrr", "nnnn", "nnnn"],
         ["nnnn", "ccgg", "rccc", "rrcc", "rggr", "nnnn"],
         ["nnnn", "grrg", "nnnn", "crgr", "ggrr", "nnnn"],
         ["nnnn", "nnnn", "grgr", "ggrr", "nnnn", "nnnn"],
         ["nnnn", "nnnn", "nnnn", "nnnn", "nnnn", "nnnn"]]
```

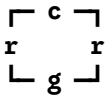
Exemple de sortie:

Exemple 1: [(1, 0), (1, 2), (2, 1)]
Exemple 2: [(1, 0), (2, 1), (2, 3), (3, 2)]
Exemple 3: [(0, 2), (0, 3), (2, 0), (2, 5), (3, 0), (3, 2), (3, 5), (4, 4), (5, 2), (5, 3)]

Explication de la première sortie:

Voici une représentation du **Board** de l'exemple 3 et de la **Tile** à placer.

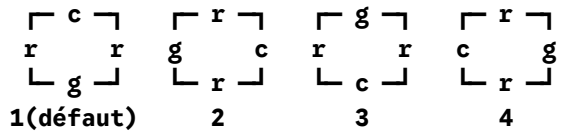




Pour trouver toutes les positions où notre tuile peut aller, nous allons procéder par élimination. Il faut d'abord identifier tous les emplacements où une tuile pourrait être placée. Les tuiles bleues représentent les tuiles déjà placées et ceux en rouge, n'ont aucune tuile qui leurs est adjacente, donc les cases vertes sont les positions valides où une tuile pourrait être placée.

0, 0	0, 1	0, 2	0, 3	0, 4	0, 5
1, 0	1, 1	1, 2	1, 3	1, 4	1, 5
2, 0	2, 1	2, 2	2, 3	2, 4	2, 5
3, 0	3, 1	3, 2	3, 3	3, 4	3, 5
4, 0	4, 1	4, 2	4, 3	4, 4	4, 5
5, 0	5, 1	5, 2	5, 3	5, 4	5, 5

Maintenant, pour chacune de ces cases, il faut regarder si la tuile peut être posée à cet emplacement et ce peu importe son orientation. Voici les 4 orientations que notre tuile peut avoir:



Pour la tuile (0, 2), l'orientation 1 est valide, donc on l'ajoute à la liste de résultats. Pour la tuile (0, 3), l'orientation 2 et 4 sont valides, donc on l'ajoute aussi à la liste des résultats. Par contre, il n'existe pas d'orientation valide pour la case (1, 1), donc on ne l'ajoute pas. On continue le processus pour chaque case jusqu'à trouver la liste finale à retourner.

0, 0	0, 1	0, 2	0, 3	0, 4	0, 5
1, 0	1, 1	1, 2	1, 3	1, 4	1, 5
2, 0	2, 1	2, 2	2, 3	2, 4	2, 5
3, 0	3, 1	3, 2	3, 3	3, 4	3, 5
4, 0	4, 1	4, 2	4, 3	4, 4	4, 5
5, 0	5, 1	5, 2	5, 3	5, 4	5, 5

Partie 5: Boggle (30 points)

Boggle est un jeu dans lequel l'objectif est de trouver le plus de mots possible sur une grille de 4x4. Le moyen de former un mot est de commencer par lettre, puis on peut sélectionner une des lettres voisines (les diagonales sont valides) et continuer de cette lettre jusqu'à avoir un mot complet. Par contre, les lettres qui ont été visitées ne peuvent pas être revisitées.

Dans une partie de boggle, tous les joueurs ont 3 minutes pour trouver le plus de mots possible chacun de leur côté. Après 3 minutes, les réponses sont comparées et seuls les mots qu'aucun des autres joueurs a trouvé compte comme des points. Par contre, il m'arrive souvent d'avoir trouvé un mot et ne plus retrouver sur la grille comment je l'ai formé.

J'ai donc besoin d'un programme qui trouve le mot sur la grille et fournit l'ordre des cases pour composer le mot. Les cases sont numérotées de 0 à 15. Les lettres ne peuvent pas être réutilisées et les les lettres qui se suivent dans le mot doivent être adjacentes sur la grille.

Spécification d'entrée et de sortie:

Vous allez recevoir une grille qui est un *array* en 2D de grandeur 4x4 contenant les lettres de la grille et vous allez aussi recevoir le mot à retrouver dans la grille.

En sortie, vous devrez donner un *array* de int de la position des lettres dans la grille pour former le mot demandé.

N.B. Tous les mots demandés auront une solution et une solution unique dans la grille.

Exemple d'entrée:

```
grid = [
['o', 'n', 'v', 'i'],
['a', 'l', 'p', 'w'],
['m', 'i', 'k', 's'],
['i', 'r', 's', 'r']]
word = "skips"

grid = [
['n', 'n', 'm', 'p'],
['o', 's', 'a', 't'],
['l', 'b', 'a', 'u'],
['i', 'z', 'f', 'n']]
word = "sablonnat"

grid = [
['s', 'a', 'a', 'u'],
['i', 'o', 't', 'e'],
['w', 'n', 'e', 'e'],
['p', 'f', 'e', 'u']]
word = "notais"
```

Exemple de sortie:

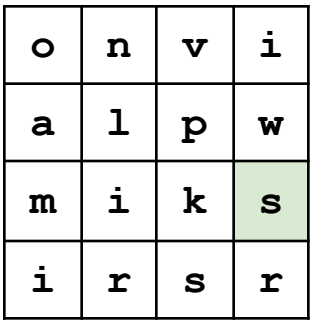
```
[14, 10, 9, 6, 11]

[5, 10, 9, 8, 4, 0, 1, 6, 7]

[9, 5, 6, 1, 4, 0]
```

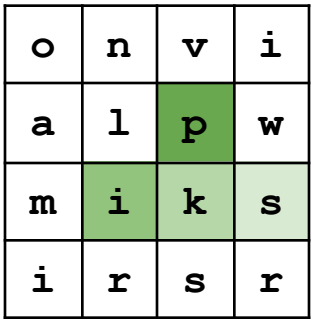
Explication de la première sortie:

On cherche à trouver le mot “skips” dans la grille suivante.
On commence donc par le premier ‘s’ à la position 11.



On prend par la suite la lettre ‘k’ à la position 10 (1 seul choix) ensuite le ‘i’ à la position 9, puis la lettre p à la position 6.

Par contre, lorsqu’on est à la lettre ‘p’, le ‘s’ voisin a déjà été utilisé dans notre mot donc nous ne pouvons pas le reprendre. Ceci est donc la mauvaise solution!!



Nous devons donc reculer dans le mot jusqu’à ce qu’une autre option soit disponible.

La seule autre décision que nous pouvons prendre de différente est de choisir le ‘s’ à la position 14 comme le début du mot. On peut maintenant aller chercher toutes les lettres suivantes aux positions 10, 9, 6 et 11 pour avoir notre solution.