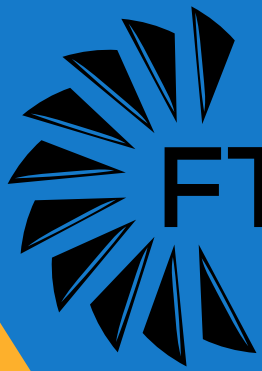
PROBLÈMES DE PROGRAMMATION
SÉNIOR DE ROBOTIQUE CRC

DÉFI SPÉCIAL

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 4% 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.

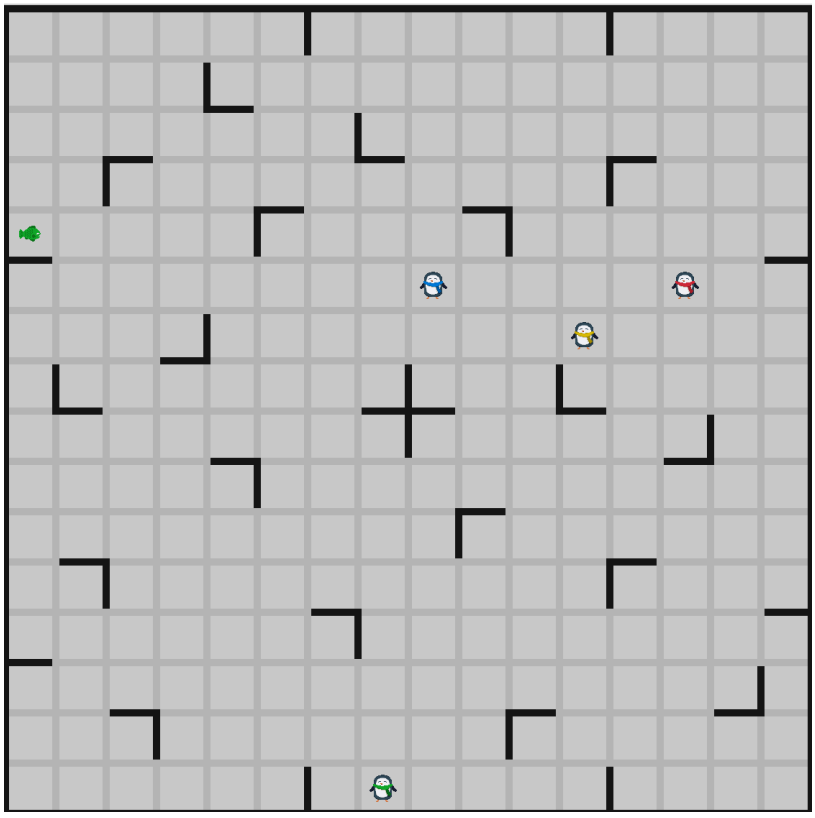
Slip N’ slide!

Dans ce problème spécial, vous devrez contrôler 4 pingouins sur une banquise afin de récupérer un poisson! Le poisson à récupérer sera seulement récupérable par un pingouin de couleur spécifique. Les pingouins devront donc travailler en équipe pour atteindre le poisson avec le moins de déplacements possible!

Comme les pingouins sont sur la banquise, ils glissent jusqu’à atteindre le prochain obstacle que ce soit un mur, ou un autre pingouin. Ainsi, vous devrez contrôler les pingouins des 4 couleurs pour que le pingouin désigné attrape le poisson en le moins de déplacements possible.

La banquise qui vous sera fournie, avec sa dimension de 16x16 cases et ces murs sont la seule banquise pour toutes les configurations sur lesquelles vous serez évalués. **Il n’y a qu’un layout de banquise pour tous les défis!** Vos pingouins pourront commencer sur n’importe quelle case de la banquise et le poisson sera toujours dans un coin de deux murs.

Pour résoudre chaque configuration, votre programme aura 10 secondes. Vous devrez donc trouver le plus rapidement possible la séquence de déplacement la plus efficace pour se rendre jusqu’au poisson!



Spécifications d’entrée et de sortie

Spécifications d’entrée

Vous recevrez en entrée la position de la position de départ des 4 pingouins sous une liste de tuples. Le tuple pour les pingouins sont la position en X (de gauche à droite, 0 à 15) puis en Y (de haut en bas, 0 à 15) et finalement la lettre en majuscule de la couleur du pingouin.

Vous recevrez aussi en entrée la position du poisson et la couleur du pingouin qui doit le récupérer dans un tuple contenant la position en X (0 à 15 de gauche à droite), la position en Y (de 0 à 15 de haut en bas) et la lettre en majuscule de la couleur.

Vous recevrez finalement un objet board de type Board pour vous aider avec les murs. Les détails de l’objet Board sont dans le fichier game_logic.py Le board reçu en entrée **n’est pas nécessaire d’être utilisé,** il sert de placement pour les murs et d’aide à l’affichage visuel lors des tests. **VOUS NE POUVEZ PAS MODIFIER UN AUTRE FICHIER QUE LE FICHIER prelim.py.** Ainsi, si vous souhaitez utiliser le Board, vous ne pouvez pas le modifier dans le fichier game_logic.py. Ceci étant dit, le board contient 33x33 tuiles pour prendre en compte la position des murs lors de l’affichage visuel. **Il est donc fortement conseillé que vous mettiez en place votre propre logique pour représenter la banquise comme elle est toujours la même dans toutes les situations à résoudre!**

Spécifications de sortie

En sortie, vous devrez fournir une série de commandes sous la forme d’une liste de string. Chaque commande est composée de 2 éléments. Le premier élément est la lettre majuscule de la couleur du pingouin à déplacer et le second élément est la lettre de la direction dans laquelle il doit glisser (haut = U, bas = D, gauche = L et droite = R).

Par exemple, si le pingouin rouge doit glisser vers la bas, on obtient la commande “RD”. La solution est donc une liste de commande pour amener le pingouin de la bonne couleur jusqu’au poisson!

Répartition des points

L'attribution des points se fait en 2 parties distinctes soit la composante de précision (nombre de déplacements de pingouin) et la composante de temps (temps avant de fournir une solution). Pour chaque configuration testée, vous aurez 50 points maximum pour votre score de précision et 20 points maximum pour votre score de vitesse. Tous les pointages seront en comparaison avec les résultats obtenus par les autres équipes pour la même configuration. Il y a donc 70 points qui sont donnés par configuration. Si votre programme ne trouve pas une solution valide dans le temps alloué, vous avez 0 points pour cette configuration. Vous serez testé sur un ensemble de 20 grilles pour faire le score du problème préliminaire.

Score de précision

Le score de précision calcule le nombre de déplacements que vos pingouins doivent faire pour que le pingouin de la bonne couleur atteigne le poisson. Si vous avez le plus petit nombre de déplacements pour résoudre la grille, vous obtenez 50 points. Pour chaque déplacement de plus que la meilleure solution trouvée, vous perdez 5 points jusqu'à atteindre 0.

Score de vitesse

Le score de vitesse calcule votre temps pour trouver une solution en comparaison avec les autres équipes. Vous avez un maximum de 20 points si vous êtes l'équipe qui trouve une solution valide le plus rapidement possible. Vous perdez 1 point par position. Seules les 20 équipes les plus rapides avec une solution valide auront des points pour le score de vitesse. (À titre de référence, 22 équipes ont soumis une solution pour le Prelim 1 ainsi la grande majorité des équipes devraient recevoir des points de vitesse). Pour recevoir des points de vitesse, votre solution fournie doit être valide.

Nombre de déplacements (meilleur trouvé = X)	Score de précision
X + 0 déplacements	50
X + 1 déplacements	45
X + 2 déplacements	40
…	…
X + 9 déplacements	5
X + 10 déplacements	0
> X + 10 déplacements	0

Vitesse par rapport aux autres équipes	Score de vitesse
plus rapide	20
2e plus rapide	19
3e plus rapide	18
…	…
19e plus rapide	2
20e plus rapide	1
21e et +	0

Comment tester votre programme?

Vous pouvez utiliser pytest comme les autres problèmes préliminaires.

Il y a 3 variables que vous pouvez contrôler. Les 2 premières sont pour la mise en place d’un `seed`. Ceci vous permet d’avoir la même configuration de défi entre les différents essais. Pour vous assurer que le seed soit actif vous devez mettre à True la variable use_seed.

La 3e variable est le nombre d'exemples par test.

Ces variables se trouvent dans le fichier test_prelim.py.