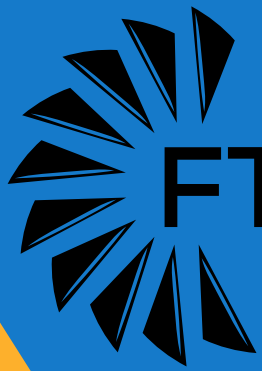
PROBLÈMES DE PROGRAMMATION
SÉNIOR DE ROBOTIQUE CRC

PRÉLIMINAIRE 3

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 23 janvier, 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.

Abeilles and bees!

Dans ce troisième problème, on se transporte dans le monde des abeilles et bees! Par contre, contrairement aux problèmes des ornithorynques, nous allons cette fois-ci avoir des problèmes qui ne sont pas pour découvrir les caractéristiques des abeilles, mais plus pour découvrir tous les jeux de mots qu'on peut faire avec abeilles et bee!

Partie 1: Fizz-Buzz-Bees (15 points)

Le jeu fizz-buzz est un classique simple qui peut facilement être joué dans un voyage en auto ou autour d'un feu de camp. Aujourd'hui nous allons lui apporter une petite modification!

Dans le jeu original de fizz-buzz, l'objectif est de compter en remplaçant les nombres divisibles par 3 par Fizz et les nombres divisible par 5 par Buzz. Nous ajoutons la règle qui si un nombre divisible par 7 est atteint, il faut remplacer le nombre par Bees!

Quand vous atteignez un nombre qui est divisible par à la fois 3 et 5, vous devez dire FizzBuzz. La même logique s'applique si vous avez un nombre divisible par 3, 5 et 7 qui serait remplacé par FizzBuzzBees.

Voici un exemple simple de la séquence originale de 1 à 10.
[“1”, “2”, “Fizz”, “4”, “Buzz”, “Fizz”, “Bees”, “8”, “Fizz”, “Buzz”]

Spécification d'entrée et de sortie:

Vous recevrez un *int* qui est le premier nombre à inclure dans la séquence et vous recevrez un second *int* qui est le dernier nombre à inclure dans la séquence.

Vous devez retourner une *list* contenant des *string* qui sont soit Fizz-Buzz-Bees ou les nombres sous forme de *string.*

Exemple d'entrée:

start = 9
end=15

start=100
end=110

start=20
end=42

Exemple de sortie:

['Fizz', 'Buzz', '11', 'Fizz', '13', 'Bees', 'FizzBuzz']

['Buzz', '101', 'Fizz', '103', '104', 'FizzBuzzBees', '106', '107', 'Fizz', '109', 'Buzz']

['Buzz', 'FizzBees', '22', '23', 'Fizz', 'Buzz', '26', 'Fizz', 'Bees', '29', 'FizzBuzz', '31', '32', 'Fizz', '34', 'BuzzBees', 'Fizz', '37', '38', 'Fizz', 'Buzz', '41', 'FizzBees']

Explication de la première sortie:

Dans le premier exemple, on compte de 9 à 15 inclusivement. Donc on commence par 9 qui est seulement divisible par 3 donc on le remplace par ‘Fizz’. Pour le nombre 10, il est seulement divisible par 5 donc on le remplace par ‘Buzz’. Pour le nombre 11, il n’est pas divisible par 3, 5 ou 7 donc on le met sous forme de string pour obtenir ‘11’. Par la suite, pour le nombre 12, il est divisible par 3 donc on le remplace par ‘Fizz’. Pour le nombre 13, il n’est pas divisible par 3, 5 et 7 donc on le convertit en string pour avoir ‘13’. Pour le nombre 14, il est divisible par 7 donc on le remplace par ‘Bees’. Finalement, le nombre 15 est divisible par 3 et 5 donc on le remplace par ‘FizzBuzz’ ce qui nous donne la séquence finale:
['Fizz', 'Buzz', '11', 'Fizz', '13', 'Bees', 'FizzBuzz']

Partie 2: Fris-bee (X points)

Des abeilles jouent au frisbee tranquillement dans le parc pendant que je marche ce matin. Comme toute personne normale qui voit des abeilles se faire des passes de frisbee je me suis demandé…

Comment fonctionne la physique d'un lancer de frisbee???

Dans mes recherches j'ai trouvé un article très intéressant du MIT (https://web.mit.edu/womens-ult/www/smite/frisbee_physics.pdf) qui simplifie la physique d'un lancer de frisbee. Nous allons nous baser sur cette ressource pour faire une mini simulation qui va montrer les coordonnées des 5 premières secondes d'un vol de frisbee selon son angle de lancer.

Voici les grands concepts du vol d'un frisbee:

Il y a 2 axes que nous voulons savoir en tout temps, la distance horizontale du point de lancer (x) et la hauteur par rapport au sol (y).

Il y a aussi 3 grandes forces qui agissent sur le frisbee. La gravité qui fait descendre le frisbee en y, le lift qui fait monter le frisbee en y et finalement nous avons le drag (friction) qui fait ralentir le frisbee en x.

Dans notre simulation, nous allons calculer à chaque 0,01 secondes le changement de position et l'enregistrer. La simulation va durer 1 seconde donc un total de 100 points de données pour modéliser la trajectoire du frisbee. Vous devrez donner des coordonnées avec une précision de 3 chiffres après la virgule, mais vous devez garder les chiffres sans arrondi pour faire tous vos calculs.

Dans tous les exemples, vous recevrez la vitesse initiale en x et en y ainsi que l'angle en degré. Toutes les autres variables seront fournies dans le fichier de code comme des constantes.

Avant de commencer la simulation, vous devrez calculer le coefficient de lift ainsi que le coefficient de drag avec les formules suivantes:

$$CL = CL0 + CLA * angle * PI/180$$

$$CD = CD0 + CDA * ((angle - ALPHA0) * PI/180)^2$$

Vous avez toutes vos variables prêtes pour commencer la simulation! On peut maintenant calculer 100 fois notre déplacement en x et en y (en l'enregistrant avec une précision de 3 décimales) pour des intervalles de 0.01 secondes.

Pour chaque itération, nous allons trouver le changement à faire (delta) pour la vitesse en x et en y. Nous allons ensuite ajuster la position en x et y. Puis nous allons enregistrer cette nouvelle position et répéter le processus!

Voici le calcul pour l'ajustement de la vitesse en x et y pour chaque itération ($\Delta t = 0.01$):

$$\Delta vy = (AIRDENSITY * vx^2 * AREA * CL * MASS/2 + G) * \Delta t$$
$$\Delta vx = -AIRDENSITY * vx^2 * AREA * CD * \Delta t$$

$$vx = vx + \Delta vx$$
$$vy = vy + \Delta vy$$
$$x = x + vx * \Delta t$$
$$y = y + vy * \Delta t$$

Une dernière règle importante pour nos calculs est que quand le frisbee touche le sol, (y<=0) on arrête de fournir la position du frisbee comme il est au sol et ne devrait plus bouger. On arrête donc la simulation et donc de fournir des coordonnées. Toutes nos coordonnées vont avoir une hauteur (y) positive (ou potentiellement 0 à cause des arrondis).

Spécification d'entrée et de sortie:

En entrée, vous recevez:

- vx0: un *float* de la vitesse initiale en x
- vy0: un *float* de la vitesse initiale en y
- angle: un *float* de l'angle de départ du frisbee en degrés

En sortie, vous devrez donner une *liste* formée de *tuples* contenant la position en x et en y sous forme de *float* avec 3 décimales de précision.

Exemple d’entrée:

```
vx0 = 16
vy0 = 0
angle = 0

vx0 = 15
vy0 = 2
angle = 5
```

Exemple de sortie:

```
[(0.16, 0.999), (0.319, 0.997), (0.479, 0.994), (0.638, 0.99), (0.798,
0.986), (0.957, 0.98), (1.115, 0.973), (1.274, 0.965), (1.433, 0.957),
(1.591, 0.947), (1.749, 0.936), (1.907, 0.925), (2.065, 0.912), (2.223,
0.899), (2.38, 0.884), (2.538, 0.869), (2.695, 0.852), (2.852, 0.835),
(3.009, 0.817), (3.165, 0.797), (3.322, 0.777), (3.478, 0.756), (3.635,
0.733), (3.791, 0.71), (3.947, 0.686), (4.102, 0.661), (4.258, 0.635),
(4.414, 0.608), (4.569, 0.58), (4.724, 0.551), (4.879, 0.521), (5.034,
0.49), (5.188, 0.458), (5.343, 0.425), (5.497, 0.392), (5.652, 0.357),
(5.806, 0.321), (5.959, 0.284), (6.113, 0.247), (6.267, 0.208), (6.42,
0.168), (6.574, 0.128), (6.727, 0.086), (6.88, 0.044), (7.032, 0.0)]

[(0.15, 1.019), (0.299, 1.037), (0.449, 1.054), (0.598, 1.07), (0.747,
1.086), (0.895, 1.1), (1.044, 1.113), (1.192, 1.126), (1.34, 1.137),
(1.487, 1.148), (1.635, 1.157), (1.782, 1.166), (1.929, 1.173), (2.076,
1.18), (2.223, 1.186), (2.369, 1.191), (2.515, 1.195), (2.661, 1.197),
(2.807, 1.199), (2.952, 1.2), (3.098, 1.2), (3.243, 1.199), (3.388,
1.197), (3.532, 1.195), (3.677, 1.191), (3.821, 1.186), (3.965, 1.18),
(4.109, 1.174), (4.252, 1.166), (4.396, 1.158), (4.539, 1.148), (4.682,
1.138), (4.824, 1.126), (4.967, 1.114), (5.109, 1.101), (5.251, 1.086),
(5.393, 1.071), (5.535, 1.055), (5.676, 1.038), (5.818, 1.02), (5.959,
1.001), (6.1, 0.981), (6.24, 0.96), (6.381, 0.938), (6.521, 0.915),
(6.661, 0.891), (6.801, 0.866), (6.941, 0.841), (7.08, 0.814), (7.219,
0.786), (7.358, 0.758), (7.497, 0.728), (7.636, 0.698), (7.774, 0.666),
(7.912, 0.634), (8.051, 0.601), (8.188, 0.566), (8.326, 0.531), (8.464,
0.495), (8.601, 0.458), (8.738, 0.419), (8.875, 0.38), (9.012, 0.34),
(9.148, 0.299), (9.284, 0.257), (9.42, 0.214), (9.556, 0.171), (9.692,
0.126), (9.828, 0.08), (9.963, 0.033)]
```

Explication de la première sortie:

Se fier à l’explication et à l’article de référence au besoin.

Partie 3: Bee-thoven (X points)

Avez-vous entendu parler du pianiste légendaire qu'est Bee-thoven ? Il paraît qu'il fait le buzz dans la ruche avec ses sérénades mielleuses.

Mais ces derniers temps, Bee-thoven est dans une impasse et ne parvient plus à composer de symphonies aussi brillantes qu'avant. Votre mission : trouver une **séquence de 4 mesures** qui respecte toutes ses exigences et lui permette de retrouver l'inspiration.

Pour chaque mesure, Bee-thoven vous fournit une **liste de groupes de notes** parmi lesquels vous devez choisir. Chaque groupe a plusieurs propriétés :

- **id** : identifiant du groupe
- **pitch** : tonalité de la mesure (un nombre entre 1 et 12)
- **duration** : durée de la mesure
- **type** : catégorie de la mesure (0, 1 ou 2)
- **higher** : si le pitch de ce groupe doit être plus haut que celui de la mesure précédente (0 ou 1)
- **forbidden** : si ce groupe ne peut pas suivre un groupe du type précisé (-1 ou le type du groupe à éviter)

Règles additionnelles à respecter :

- Vous devez choisir **un seul groupe par mesure**.
- Deux groupes consécutifs **ne peuvent pas être du même type**.
- La **somme des durées** des 4 mesures doit être **exactement 10**.
- Les contraintes **higher** et **forbidden** doivent être respectées.

Spécification d'entrée et de sortie:

En entrée vous recevrez une ***list* de 4 mesures**. Chaque mesure est elle-même une ***list*** des groupes de notes possibles de choisir pour cette mesure. Chaque groupe est représenté par un ***tuple*** sous la forme : **`(id, pitch, duration, type, higher, forbidden).`**

En sortie, vous devrez retourner une ***list* des IDs** correspondant à la séquence correcte de groupes qui satisfait les exigences de Bee-thoven.

Exemples d'entrée:

```
1. [[(0, 5, 2, 1, 0, -1), (1, 3, 3, 0, 0, -1)],
    [(10, 4, 3, 1, 1, -1), (11, 7, 1, 2, 0, 0)],
    [(20, 3, 3, 1, 0, -1), (21, 8, 4, 2, 0, 1), (22, 6, 2, 0, 1, -1)],
    [(30, 9, 2, 2, 0, -1), (31, 4, 5, 0, 0, -1)]]
```

```
2. [[(0, 4, 3, 0, 0, -1), (1, 6, 4, 1, 0, -1)],
    [(10, 7, 2, 1, 0, -1), (11, 5, 3, 2, 0, -1)],
    [(20, 8, 3, 2, 0, -1), (21, 3, 4, 0, 0, -1)],
    [(30, 9, 2, 0, 0, -1), (31, 2, 5, 1, 0, -1)]]

3. [[(0, 6, 3, 1, 0, -1), (1, 3, 2, 0, 0, -1)],
    [(10, 5, 2, 2, 1, -1), (11, 4, 4, 1, 0, 0)],
    [(20, 7, 5, 0, 1, -1), (21, 2, 3, 2, 0, 1), (22, 9, 3, 1, 0, -1)],
    [(30, 4, 7, 0, 0, -1), (31, 10, 3, 2, 0, -1)]]
```

Exemples de sortie:

```
1. [1, 10, 22, 30]
2. [0, 10, 20, 30]
3. [1, 10, 22, 31]
```

Explication de la première sortie:

Pour la **première mesure**, les deux groupes n'ont aucune contrainte, donc les deux peuvent être choisis.
Possibilités: [0, …, …, …] ou [1, …, …, …]

Pour la **deuxième mesure**, le groupe 10 doit avoir un pitch plus grand que la mesure précédente, donc il peut seulement suivre le groupe 1 (4 > 3). Le groupe 11, lui, ne peut pas suivre un groupe de type 0, donc il peut seulement suivre le groupe 0.
Possibilités: [0, 11, …, …] ou [1, 10, …, …]
Pour la **troisième mesure,** seulement le groupe 22 peut être placé après le groupe 10, car le groupe 20 est du même type (type 1), le groupe 21 est interdit de suivre un groupe de type 1 et le groupe 22 a un pitch plus haut que celui de la mesure précédente (6 > 4). Par la même logique, seulement le groupe 20 peut être placé après le groupe 11, car le groupe 21 est du même type (type 2) et le groupe 22 a un pitch plus petit que la mesure précédente (6 < 7).
Possibilités: [0, 11, 20, …] ou [1, 10, 22, …]

Pour la **quatrième mesure**, on doit atteindre une somme des durées égale à 10 et il n'y a qu'une seule combinaison de groupes qui respecte toutes les règles et la durée : **[1, 10, 22, 30]**.

Partie 4: bee-p bee-p! (X points)

Une abeille doit se rendre à sa maison, mais il y a beaucoup d'autres abeilles doivent aussi se rendre chez elles. Aide cette abeille à se rendre le plus rapidement chez elle.

Avec les données présentées, déterminer le chemin le plus rapidement

Spécification d'entrée et de sortie:

En entrée, vous aurez

r = un *int* du nombre de route possible

v = une *liste* de *int* de la vitesse limite en m/h par route

t = une *liste* de *int* de la vitesse du trafic en m/h par route

trafic_distance = une *liste* de *int* de la longueur en m du trafic par route

d = une *liste* de *int* de la distance à parcourir par route

En sortie, vous devez donner un *int* du numéro de la route la plus rapide

Exemple d'entrée:

r = 2
v = (50, 40)
t = (15, 25)
trafic_distance = (15, 4)
d = (30, 25)

r = 3
v = (60, 30, 25)
t = (20, 25, 24)
trafic_distance = (30, 40, 40)
d = (45, 50, 47)

r = 3
v = (30, 30, 30)
t = (13, 15, 12)
trafic_distance = (15, 12, 13)
d = (30, 30, 30)

Exemple de sortie:

2

1

2

Explication de la première sortie:

La route 1 a 15 mètres de trafic qui va à 15m/h et 15 mètres à 50 m/h ce qui fait que ça prend 1h dans le trafic et 18 minutes hors du trafic. Le temps est calculé en divisant la distance par la vitesse, et multiplié par 60 pour le temps en minutes. La route 2 a 4 mètres à 25 m/h ce qui prend 9 minutes 36s et 21 mètres à 40 m/h prend 31 minutes 30s, donc la route 2 prend moins de temps!

Partie 5: Ab-eille ça fait mal (X points)

OUCH! Une abeille vous a piqué et vous voulez savoir quel anti venin / anti-douleur vous devriez prendre pour aider à soulager la douleur et contrer l'effet de la piqûre. Vous devrez utiliser le niveau de douleur ressenti, l'endroit géographique où vous vous êtes fait piquer, ainsi que la famille taxonomique et le type de ruche de l'abeille pour déterminer quelle sorte d'abeille vous a piqué. Ensuite, vous pourrez décider quel remède utiliser pour atténuer la douleur et l'inflammation. Voici la liste des différents types d'abeille qui pourraient vous avoir atteint avec leur dart:

Sorte d'abeille	Niveau de douleur	Endroit géographique	Famille taxonomique	Type de ruche
sweat bee	4	afrique, asie, europe	halictidae	sol, arbre
killer bee	8	amérique du nord, amérique du sud, afrique	apidae	sol, trou
carpenter bee	5	océanie, europe	apidae	bois mort, bambou
honey bee	4	amérique du nord, amérique du sud, asie, europe	apidae	trou, ruche artificielle
bumblebee	2	amérique du nord, amérique du sud, asie, europe	apidae	trou, arbre, jardin
dark bee	8	europe, amérique du nord	apidae	arbre
buckfast bee	6	europe, afrique	apidae	ruche artificielle
mining bee	6	amérique du nord, afrique, europe, asie	andrenidae	sol
mason bee	4	asie, océanie	megachilidae	roche, arbre, bambou
resin bee	5	europe	megachilidae	ruche artificielle, arbre

Les remèdes possibles sont dans la liste suivante:

- épinéphrine (sweat bee, carpenter bee, bumblebee, mining bee)
- antihistaminique (killer bee, resin bee, dark bee, buckfast bee)
- glace (mason bee, honey bee, bumblebee)
- crème antiseptique (killer bee, dark bee, resin bee)

***La liste de remède est semi-représentative de la réalité. Ne pas se fier à ce problème de programmation si vous vous faites piquer par une abeille.

Spécification d'entrée et de sortie:

En entrée, vous recevrez le niveau de douleur ressenti, l'endroit géographique où vous étiez lors de l'incident, la famille taxonomique de l'abeille et le type de ruche de l'abeille.
pain: int
region: str
family: str
nest_type: str
En sortie, vous devrez donner une *list[str]* qui contient 1 à x remèdes à utiliser

Exemple d'entrée:

```
pain=4 geography=europe family=halictidae nest=ground
pain=2 geography=asia family=apidae nest=hole
pain=8 geography=africa family=apidae nest=ground
```

Exemple de sortie:

```
[epinephrine]
[epinephrine, ice]
```
[antihistamine, antiseptic cream]

Explication de la première sortie:

La seule sorte d'abeille qui répond à tous les critères donnés en entrée est la "sweat bee" et le remède est l'épinéphrine.