POLYNÉSIE – 2025 – sujet1

Exercice 2 (6 points)

Cet exercice porte sur la programmation Python et la récursivité.

Le jeu du baguenaudier est un jeu de casse-tête constitué d'une réglette comportant `n` cases, numérotées de 1 à `n`.

Chaque case peut être soit vide, soit contenir un pion.

« Jouer » une case consiste à placer un pion dans la case (remplir) si elle est vide ou enlever un pion (vider) si elle est remplie.

Initialement, toutes les cases sont vides.

Le but du jeu est de remplir toutes les cases du baguenaudier en suivant les règles suivantes :

- on ne peut jouer qu'une case à la fois ;
- chaque case ne peut contenir qu'un pion ;
- on peut toujours jouer la case 1 ;
- si le baguenaudier n'est ni vide ni rempli, on peut aussi jouer la case qui suit la première case remplie ;
- aucune autre case ne peut être jouée.

Exemple de situation avec un baguenaudier de 5 cases.

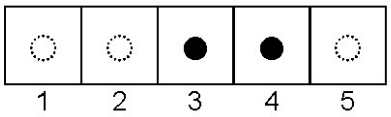

Figure 1. Situation baguenaudier 5 cases

Dans cette situation, on peut poser un pion dans la case 1 ou enlever le pion de la case 4 mais on ne peut pas jouer les cases 2, 3 et 5.

1. On considère un baguenaudier à 4 cases sur lequel les trois dernières cases sont remplies.

   Désigner les cases que l'on peut jouer et dessiner l'état du baguenaudier à la suite de chacun des coups possibles.

Pour modéliser le baguenaudier on utilise un tableau (de type `list`) de booléens. Une case vide est représentée par `False` et une case remplie par `True`.

L'exemple de situation du baguenaudier de 5 cases de la Figure 1 présentée plus haut sera ainsi représenté par le tableau suivant.

```
[False, False, True, True, False]
```

La fonction `initialiser` prend en paramètre un nombre entier `n` et elle renvoie une liste modélisant un baguenaudier vide de `n` cases.

Exemple :

```
>>> initialiser(3)
[False, False, False]
```

2. Écrire le code de la fonction `initialiser`

La fonction `victoire` renvoie un booléen indiquant si toutes les cases du baguenaudier sont remplies.

```
1 def victoire(tab):
2     for etat_case in tab:
3         if etat_case == ...:
4             return ...
5     ...
```

3. Recopier et compléter les lignes 3, 4 et 5 du code de la fonction `victoire`.

La fonction `indice_premiere_case_occupee` prend en paramètre un tableau `tab` modélisant l'état du baguenaudier et renvoie l'indice de la première case remplie du baguenaudier, ou `None` si le baguenaudier est vide.

Exemples :

```
>>> indice_premiere_case_occupee([True, True, True])
0
>>> indice_premiere_case_occupee([False, False, False, True,
True])
3
>>> indice_premiere_case_occupee([False, False, False, False])
# pas de sortie car valeur None
```

4. Écrire le code de la fonction `indice_premiere_case_occupee`.

La fonction `coup_valide` prend en paramètres `tab` qui est une liste modélisant l'état d'un baguenaudier, et un entier `case` qui est l'indice de la case à jouer. La fonction renvoie `True` si le coup respecte les règles et `False` si ce n'est pas le cas.

L'indice de la première case du jeu est 0. Pour que le coup soit valide, il faut aussi s'assurer que l'indice est un entier positif inférieur à la longueur de `tab`.

5. Écrire le code la fonction `coup_valide`.

La fonction `changer_case` prend en paramètres un tableau `tab` modélisant l'état du baguenaudier et un indice de case nommé `case`. Si le coup désigné par `case` est valide, la fonction renvoie le tableau modélisant le baguenaudier obtenu après avoir changé l'état de la case correspondante. Si le coup joué n'est pas valide, le baguenaudier passé en paramètre est renvoyé sans modification.

Exemples :

```
>>> changer_case([False, False, False], 1) # coup non valide
[False, False, False]
>>> changer_case([False, False, False], 0) # coup valide
[True, False, False]
>>> changer_case([False, True, False], 2)  # coup valide
[False, True, True]
```

6. Écrire le code de la fonction `changer_case`.

La résolution d'un baguenaudier de 3 cases, en Figure 2 et 4 cases, en Figure 3 sont données ci-dessous.

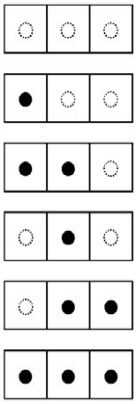

Figure 2. Résolution du baguenaudier à 3 cases

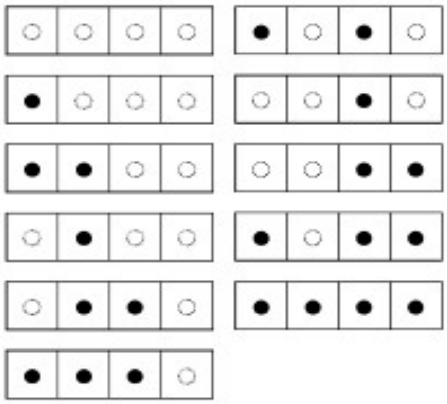

Figure 3. Résolution du baguenaudier à 4 cases

Il est possible d'obtenir la solution du jeu du baguenaudier, sous forme d'affichage, en utilisant des fonctions récursives nommées `vider` et `remplir`. Ces deux fonctions prennent en paramètre un entier `n` correspondant au nombre de cases du jeu. La fonction `vider` vide un baguenaudier de `n` cases initialement remplies. La fonction `remplir` remplit un baguenaudier de `n` cases initialement vides.

Le code de la fonction `vider`, incomplet, est donné ci-dessous. On notera que les cases sont numérotées à partir de 1 dans la suite et que l'appel `vider(0)` ne fait rien.

```
1 def vider(n):
2     if n == 1:
3          print('Vider case 1')
4      elif n > 1:
5          vider(n-2)
6          print(...)
7          remplir(n-2)
8          vider(n-1)
```

7. Recopier et compléter la ligne 6 du code de la fonction `vider` pour qu'elle affiche le texte `'Vider case '` suivi de la valeur de l'entier `n`.

8. En supposant que l'appel `remplir(1)` affiche `' Remplir case 1'` et que l'appel `remplir(0)` ne fait rien, donner les affichages, dans la console, produits par l'exécution de `vider(3)`.

9. En vous inspirant de la fonction `vider`, de la résolution donnée à la question précédente pour n = 3 et des Figures 2 et 3, écrire le code de la fonction récursive `remplir`.

10. On envisage d'utiliser la fonction `vider` pour un baguenaudier de 2000 cases. Expliquer si ce code est adapté à un baguenaudier de si grande taille. Justifier votre réponse.