Exercice 1

jeu de Dominos

1.1 version classique

Le jeu de *Dominos* est un jeu très simple, ou, pour gagner, il faut être le premier joueur à avoir posé tous ses dominos. Une fois le premier domino placé sur la table, le joueur suivant doit à son tour poser un domino ayant le même nombre de points sur au moins un côté du domino précédemment posé.

Un domino est constitué de côtés, droite/gauche ou haut/bas selon comment la pièce sera disposée.

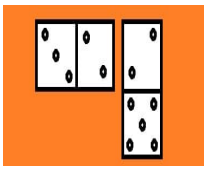

FIGURE 1 – début de partie

La disposition importe peu : il faut que la chaine reste ouverte.

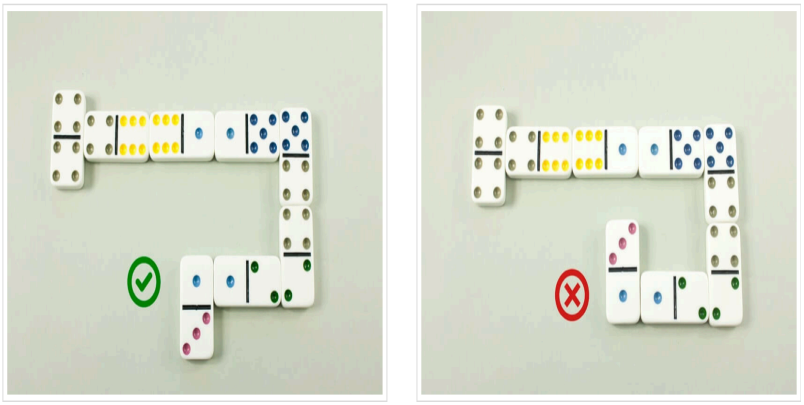

FIGURE 2 – Exemple de disposition juste (à gauche) et fausse (à droite)

On utilise la définition de classes suivantes :

```
class Domino:
    def __init__(self,val1,val2):
        self.val1 = val1
        self.val2 = val2
        self.suiv = None


class Partie:
    def __init__(self,first):
        self.first = first

    def last(self):
        M = self.first
        while not M.suiv is None:
            M = M.suiv
        return '{}:{}'.format(M.val1,M.val2)

```

```
18      def atteindre_domi(self,val1,val2):
19          D = self.first
20          while not D.suiv is None and (D.val1,D.val2) != (val1,val2):
21              D = D.suiv
22          if (D.val1,D.val2) == (val1,val2):
23              return D
24          else:
25              return self.first
26
27      def inserer(self,D_place,D_a_inserer):
28              #à completer
29
30      def __repr__(self):
31          M = self.first
32          s = '{}:{} '.format(M.val1,M.val2)
33              # à completer
34          return s
```

Qu a. On cherche à représenter la partie de l'image de gauche (voir plus haut). Les dominos seront instanciés à l'aide des noms D1, D2, D3, ... Ecrire les instructions qui instancient tous les dominos de la partie, avec, pour chacun, leurs valeurs et le domino suivant.

Qu b. Ecrire l'instruction qui doit créer l'objet `partie1` à partir de ce plateau de jeu. (classe `Partie`)

Qu c. Compléter la méthode de classe `__repr__` qui surcharge la fonction `print`

Qu d. Commenter la méthode de classe `atteindre_domi`. A quoi sert-elle ? Quelle est sa complexité asymtotique ?

Qu e. Imaginons que l'état de la partie soit celui-ci :

```
print(partie1)
4:4 => 4:6 => 6:1 => 1:5 => 5:4 => 4:2 => 2:1 => 1:3
```

Compléter la méthode de classe **`inserer`** qui permet d'insérer un domino (double) dans la chaine de dominos à partir des instructions suivantes :

```
D = partie1.atteindre_domi(4,2)
D10 = Domino(2,2)
partie1.inserer(D,D10)
```

Le nouvel état de la partie devrait alors être :

```
print(partie1)
4:4 => 4:6 => 6:1 => 1:5 => 5:4 => 4:2 => 2:2 => 2:1 => 1:3
```

Qu f. Insérer le domino `1:1` à sa place, dans la partie.

Qu g. Ajouter une méthode de classe `poser` qui pose un domino à la suite du dernier domino posé dans la partie, à condition que celui-ci soit bien en correspondance. La fonction va alors retourner l'état de la partie (si le nouveau domino peut être posé), ou bien un message signifiant que la pose est interdite. Tester votre methode de classe `poser` en choisissant un domino correct, puis un domino non correct.