Corrigé — Bataille

I Consignes

II Algorithme Classiques

Question 1. *Recherche dichotomique dans une liste triée*

Question 2. *Copie d'une matrice*

III Bataille

Question 3. *Valeur des cartes*

coeff	parcours	test	résultat
2	3	3	3

```
def valeur(c):
    for i in range(len(s_valeurs)):
        if s_valeurs[i] == c:
            return i
    return -1
```

Question 4. *Bataille*

coeff	branchement	valeurs
2	6	3

```
def bataille(c1, c2):
    v1, v2 = valeur(c1[0]), valeur(c2[0])
    if v1 > v2:
        return 1
    elif v1 < v2:
        return 2
    else:
        return 0
```

Question 5. *Mélange du paquet*

coeff	import	parcours	création	mélange
3	2	3	2	2

```
from random import shuffle

def paquet():
```

```
    cartes = []
    for col in s_couleurs:
        for c in s_valeurs:
            cartes.append((c,col))
    shuffle(cartes)
    return cartes

# ou

def paquet():
    cartes = [ (c, col) for col in s_couleurs for c in s_valeurs ]
    shuffle(cartes)
    return cartes
```

Question 6. *Vérification du mélange*

coeff	taille	parcours	doublon ?	booléen
2	1	2	4	2

```
def verification(p):
    if len(p) != 52:
        return False
    cartes = {}
    for c in p:
        if c in cartes:
            return False
        cartes[c] = 1
    return True
```

Question 7. *Désérialisation*

coeff	parcours	ajouts	vérification
2	2	5	2

```
def chargement(s):
    p = []
    for i in range(0, len(s), 3):
        p.append((s[i], s[i+1:i+3]))
    if verification(p):
        return p
```

Question 8. *Distribution*

coeff	assertion	parcours	répartition	retour
2	1	2	4	2

```python
def distribution(p):
    assert len(p) == 52
    alice = []
    bob = []
    for i in range(len(p), 2):
        alice.append(p[i])
        bob.append(p[i+1])
    return alice, bob
```

Question 9. *Tour de jeu*

coeff	centre	tas vide	égalité	tas finaux
4	2	2	2	3

```python
def tour(tas_alice, tas_bob):
    centre = []
    res = -1 # Résultat de la dernière bataille. -1 signifie en attente.
    while res < 1:
        # Alice place au centre si elle peut
        if len(tas_alice) == 0:
            return tas_alice, (centre + tas_bob)
        c1 = tas_alice.pop()
        centre.append(c1)
        # Bob place au centre si elle peut
        if len(tas_bob) == 0:
            return (centre + tas_alice), tas_bob
        c2 = tas_bob.pop()
        centre.append(c2)
        # Resolution si en attente
        if res == -1 :
            res = bataille(c1, c2)
        else: # Sinon passage en attente.
            res = -1
    if res == 1: # Alice a gagné
        return (centre + tas_alice), tas_bob
    else: # Bob a gagné
        return tas_alice, (centre + tas_bob)
```

Question 10. *Partie*

coeff	initialisation	boucle	résultats
3	2	4	3

```python
def partie():
    p = paquet()
```

```
    tas_alice, tas_bob = distribution(p)
    nb_tour = 0
    while len(tas_alice) > 0 and len(tas_bob) > 0:
        tas_alice, tas_bob = tour(tas_alice, tas_bob)
        nb_tour += 1
    if len(tas_alice) > 0 :
        return 1, nb_tour
    else:
        return 2, nb_tour
```