

Algorithmique des tableaux

Compétences

- Itérer dans un tableau avec les structures de contrôle for/while
- Modifier les données d'un tableau

Contexte : Le jeu de YAMS

Le Yams se joue avec 5 dés et se finit une fois toutes les cases de la fiche de score remplies. Chaque joueur joue tout à tour et dispose de 3 lancers à chaque coup. L'objectif étant de réaliser des combinaisons qui rapportent des points. Le joueur à le choix de reprendre tout ou une partie des dés à chaque lancer, selon son gré, pour tenter d'obtenir la combinaison voulue. À chaque tour, le joueur doit obligatoirement inscrire son score dans une des cases de la feuille de marque que ce soit par un X ou par les points qu'il a obtenus.

Il peut arriver lors d'un tour que le résultat ne convienne pas au joueur et qu'il se dise qu'il pourrait faire un plus grand score sur un autre tour. Il peut alors choisir de barrer une autre case à la place. Bien entendu, il ne pourra plus faire cette combinaison par la suite.

Lorsque le total intermédiaire est égal ou supérieur à 63 points, un bonus de 37 points supplémentaires est accordé, ce qui peut faire la différence au décompte final. Soyez donc stratégique!

Yam's Feuille de Score

JOUEURS			
Total de 1			
Total de 2			
Total de 3			
Total de 4			
Total de 5			
Total de 6			
Total			
Si total I > 63 alors Bonus de 35 points			
Total partie intermédiaire			
Brelan (Total des 3 dés)			
Carré (Total des 4 dés)			
Full (25 points)			
Petite Suite (30 points)			
Grande Suite (40 points)			
Yams (50 points)			
Chance (Total des 5 dés)			
Total II			
TOTAL			

Le programme à réaliser

Pour ce TP nous utiliserons le type natif array proposé par Python et les nombres sont des entiers courts non signés sur 2 octets.

Exemple :

```
from array import *
my_array = array('H', [2, 7, 3])
```

Faut un début à tout...

1. Écrire une fonction `rollDiceString(n)` permettant d'obtenir un tirage de n dés sous la forme d'une chaîne de caractères. Les valeurs sont séparées par des espaces.

Exemple :

```
rollDiceString(5)
6 6 5 1 3
```

2. Écrire une fonction équivalente `rollDice(n)` mais qui donne le tirage sous la forme d'un tableau contenant des valeurs entières.

Exemple :

```
rollDice(5)
[3, 6, 4, 3, 2]
```

3. Écrire une fonction `sumValues(t, n)` qui renvoie la somme des valeurs affichées par les n dés sans utiliser la fonction Python `sum`.

Exemple :

```
dices = rollDice(5)
```

```
sumValues(dices, 5)
18
```

4. Écrire une fonction `numberOfDiceWithSameValue(t, n, value)` qui renvoie le nombre de dés affichant la valeur passée en paramètre.

Exemple :

```
numberOfDiceWithSameValue(dices, 5, 3)
2
```

5. Écrire une fonction `histogram(t, n)` qui renvoie un tableau contenant le nombre de dés identiques pour chaque face. La première case du tableau stocke le nombre de dés.

Exemple :

```
histogram([3, 6, 4, 3, 2], 5)
[5, 0, 1, 2, 1, 0, 1]
```

6. Écrire un script utilisant la fonction `histogram` afin d'obtenir l'histogramme pour un nombre de tirage n . Que doit-on vérifier si n est grand?

7. Écrire une fonction `maxSubarraySum(t, n)` renvoyant la somme des faces pour le plus grand nombre de dés identiques dans le tirage. S'il y a égalité la fonction renvoie la plus grande somme.

Exemple :

```
maxSubarraySum([5, 6, 5, 3, 6], 5)
12
```

Passons à la suite ...

8. Écrire une fonction `moveDice(t, n, value)` qui renvoie un tableau avec les dés correspondants à la valeur passée en paramètre, placés au début du tableau sans changer l'ordre respectif des autres dés.

Exemple :

```
moveDice([5, 6, 5, 3, 6], 5, 6)
[6, 6, 5, 5, 3]
```

9. Écrire une fonction `rollDiceAgain(t, nt, s, ns)` permettant de ne relancer qu'une partie des dés. Cette fonction possède 4 paramètres un tableau du tirage, le nombre d'éléments dans tirage, un tableau des indices des dés à relancer, le nombre d'éléments de ce dernier et renvoie le nouveau tirage.

Exemple :

```
rollDiceAgain([4, 4, 4, 1, 2], 5, [3, 4], 2)
# Résultat
[4, 4, 4, 5, 3]
```

10. Écrire une fonction `searchSequence(t, n)` permettant de détecter la présence d'une suite c'est à dire la combinaison de dés avec des valeurs qui se suivent. La fonction renvoie 0 si la suite est [1, 2, 3, 4, 5], 1 pour [2, 3, 4, 5, 6] et -1 si aucune des deux suites n'est détectées. Attention les valeurs des dés peuvent être rangées dans n'importe quel ordre au moment du tirage.

Exemple :

```
searchSequence([4, 6, 3, 2, 5], 5)
# Résultat :
1
```

11. Écrire une fonction permettant de lancer tout ou une partie des dés trois fois (ou moins si toutes les faces de dés sont identiques) pour obtenir le plus grand nombre de dés identiques possible. En cas d'égalité la fonction choisie les dés avec la valeur de face la plus grande. Cette fonction prendra en paramètre un tableau des cases encore disponibles dans la partie supérieure de la grille. Cette fonction renvoie un tableau du tirage obtenu à chaque lancer.

Exemple :

```
#Résultat du script
jouerTour(array('H', [3, 4, 6]))
tirage [4 4 5 1 6] tour 1
tirage [4 4 4 4 1] tour 2
tirage [4 4 4 4 4] tour 3
```

```
jouerTour(array('H', [3, 4, 6]))
tirage [3 3 2 2 1] tour 1
tirage [4 4 3 1 3] tour 2
tirage [4 4 1 2 6] tour 3
```

```
jouerTour(array('H', [1, 2, 6]))
tirage [1 1 1 1 4] tour 1
tirage [1 1 1 1 1] tour 2
[1 1 1 1 1]
```

Exercice complémentaire

En modifiant la fonction `isSection` de l'exercice 15 de la feuille du TD1, écrire une nouvelle fonction `numberSections(t, nt, s, ns)` qui calcule et retourne le nombre de fois où le tableau non vide `s` apparaît comme section du tableau `t`.

Par exemple, le tableau [1, 2] correspond à deux sections du tableau [5, 1, 2, 3, 1, 2, 1]. Autre exemple, le tableau [1, 2, 1] correspond à deux sections du tableau [5, 1, 2, 1, 2, 1]. Dans ce dernier cas, les deux sections se chevauchent : [5, 1, 2, 1, 2, 1] et [5, 1, 2, 1, 2, 1].