INF1 - TP 06 Exercices avec des tableaux

UNIVERSITÉ DE RENNES 1

Exercice 1 : Statistiques

Étant donné une série statistique $(x_1, x_2, \ldots, x_n)$, on définit la moyenne de cette série comme

$$m = \frac{1}{n} \sum_{i=1}^{n} x_i$$

1.a] Écrire une fonction `moyenne` qui prend en paramètre un tableau d'entiers, et qui calcule et renvoie la moyenne arithmétique des valeurs du tableau sous forme de flottant. Par exemple, `moyenne({3,7,6,18]}` renvoie `8.5`.

Pour la même série statistique, et m la moyenne de cette série, on définit la variance de cette série comme

$$\sigma = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - m)^2$$

1.b] Écrire une fonction `variance` qui prend en paramètre un tableau d'entiers, et qui calcule et renvoie la variance des valeurs du tableau sous forme de flottant. Par exemple, `variance({3,7,6,18})` renvoie `43.0`.

Exercice 2 : Histogramme - Tableaux de notes

```
 0 *
 1 *
 2
 3
 4 **
 5 *
 6 **
 7 *
 8 **
 9 ****
10 **
11 ***
12 ***
13 **
14 *
15
16 *
17 *
18
19 *
20
```

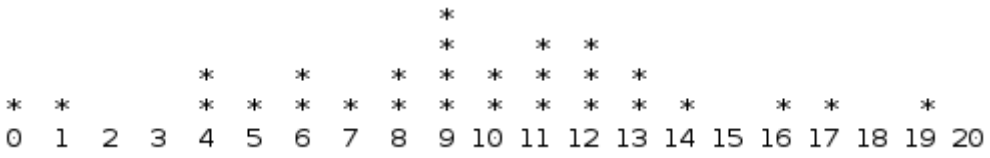
Dans cet exercice, on manipule des tableaux d'entiers de longueur quelconque qui contiennent des notes comprises entre 0 et 20. Par exemple, `int[] notes = {6, 11, 13, 16, 12, 8, 9, 7, 12, 9, 8, 10, 14, 11, 9, 1, 6, 4, 19, 17, 12, 13, 4, 0, 5, 9, 10, 11};`

Le but de cet exercice est de calculer et d'afficher un histogramme des notes. Un histogramme est un nouveau tableau donnant le nombre d'occurrences de chaque valeur possible, c'est-à-dire ici un histogramme est un tableau de 21 entiers dont la k-ième case contient le nombre de fois où la note k a été donnée.

2.a] Écrire une fonction `calculHistogramme` qui prend en paramètre un tableau `tab` et qui calcule et renvoie l'histogramme associé. Par exemple, `calculHistogramme(notes)` renvoie le tableau {`1, 1, 0, 0, 2, 1, 2, 1, 2, 4, 2, 3, 3, 2, 1, 0, 1, 1, 0, 1, 0`}.

2.b] Écrire une fonction `afficheHistoHorizontal` qui prend un histogramme `tab` en paramètre et l'affiche par lignes. Par exemple, si on utilise cette fonction sur l'histogramme précédent, on obtient l'affichage ci-contre.

2.c] **(Challenge)** Écrire une fonction `afficheHistoVertical` qui prend un histogramme `tab` en paramètre et l'affiche par colonnes. Par exemple, si on utilise cette fonction sur l'histogramme précédent, on obtient l'affichage suivant



Exercice 3 : Polynômes

3.a] Écrire une fonction `affichePoly` qui prend en paramètre un tableau `poly` à valeurs de type `double` et qui l'affiche sous le format d'un polynôme en rajoutant les puissances de `x`. Par exemple, `affichePoly({5.5, 7.7, 3.3, 2.2})` sera affiché de la manière suivante `5.5 + 7.7x + 3.3x^2 + 2.2x^3`.

3.b] Écrire une fonction `evaluePoly` qui prend en paramètres un tableau `poly` et un `double` `y` et renvoie l'évaluation du polynôme représenté par le tableau `poly` en la valeur `y`. Par exemple, `evaluePoly({5.5, 7.7, 3.3, 2.2},2.0)` calculera $5.5 + 7.7 \times 2.0 + 3.3 \times 2.0^2 + 2.2 \times 2.0^3$ et renverra donc `51.7`.

3.c] Écrire une fonction `additionPoly` qui prend en paramètres deux tableaux `poly1` et `poly2` représentant deux polynômes de degrés quelconques et qui renvoie un nouveau tableau qui représente l'addition des deux polynômes passés en paramètres. Par exemple, `additionPoly({5.5, 7.7, 3.3, 2.2},{3.87, 0, 12.84})` renverra {`9.37, 7.7, 16.14, 2.2`}.

3.d] **(Challenge)** Écrire une fonction `deriveePoly` qui prend en paramètre un tableau `poly` et qui renvoie un nouveau tableau qui correspond à la dérivée de ce polynôme. On rappelle que la dérivée de cx^{n+1} est $c(n+1)x^n$. Par exemple, `deriveePoly({5.5, 7.7, 3.3, 2.3})` renvoie {`7.7, 6.6, 6.9`}.

3.e] **(Challenge)** Écrire une fonction `multiplicationPoly` qui prend en paramètres deux tableaux `poly1` et `poly2` représentant deux polynômes de degrés quelconques et qui renvoie un nouveau tableau qui représente la multiplication des deux polynômes passés en paramètres. Par exemple, `multiplicationPoly({5.5, 7.7, 3.3, 2.2},{3.8, 0, 12.84})` renvoie {`20.9, 29.26, 83.16, 107.228, 42.372, 28.248`}.
Indication : on peut commencer par écrire deux fonctions auxiliaires :

- `multiplicationScalaire` : multiplie chaque coefficient par un `double a`, par exemple `multiplicationScalaire({1.0, 2.0, 3.4}, 2.0)` renvoie {`2.0, 4.0, 6.8`}
- `multiplicationPuissanceX` : multiplie le polynôme par x^a où `a` est un entier passé en paramètre, par exemple, `multiplicationPuissanceX({1.0, 2.0, 3.4}, 2)` renvoie {`0.0, 0.0, 1.0, 2.0, 3.4`}