PROG1 : Travaux dirigés 3

les tableaux et les strings

Exercice 1:

Créez une fonction *getMinMaxInteger* qui va prendre en paramètre un tableau d'entiers, et qui va retourner deux nombres dans un tableau : le premier le minimum, le deuxième le maximum.

Exemple d'entrée : getMinMaxInteger([1,7,6,9,14,0])
Sortie attendue : [0,14]
Exemple d'entrée : getMinMaxInteger([0,0,0,0,0])
Sortie attendue : [0, 0]

Exercice 2:

2.1. Créez une fonction *getIntegersBetween* qui va retourner dans un tableau les nombres entiers dans une intervalle donnée, les deux bornes (inférieure et supérieure) sont exclues.

Exemple d'entrée : getIntegersBetween(4,10)
Sortie attendue : [5,6,7,8,9]
Exemple d'entrée : getIntegersBetween(5,5)
Sortie attendue : [ ]

2.2. Refaites cet exercice avec une fonction récursive. Vous devez vous poser deux questions :

- Quel est la terminaison de récursivité ?
- Quel est l'appel récursif de la fonction ?

Exercice 3 :

On souhaite créer un outil statistique pour calculer des indicateurs basiques tels que la variance et l'écart-type. Pour cela créez les fonctions suivantes :

- La fonction *sum* qui, étant donné un tableau de nombres, retourne la somme des éléments de cette liste, ou 0 si la liste est vide
- la fonction mean qui, étant donné un tableau non vide de nombres, retourne la moyenne des éléments de cette liste
- la fonction *square* qui, étant donné un tableau de nombres L, retourne la liste des carrés des éléments de L
- la fonction *variance* qui étant donné un tableau non vide de nombres, renvoie la variance (la différence entre la moyenne des carrés des éléments de la liste, et le carré de la moyenne des éléments de la liste)
- la fonction *standardDeviation* qui étant donné une liste non vide de nombres, renvoie l'écart-type de la liste (racine carrée de la variance)

Quelques exemples d'entrée-sortie :

- sum([1,3,7,6,10]) = 27;
- sum([]) = 0
- mean([12, 14,17]) = 14.333333
- square([3,8,5]) = [9,64, 25]
- variance([1,9,7]) = 11.555555555556
- standardDeviation([1,9,7]) = 3.399

Exercice 4 :

Vous allez créer une fonction *mapIt* qui va transformer le contenu d'un tableau hétérogène composé d'int et de string :

- pour le cas d'un entier : on le remplacera par son carré;
- pour le cas d'un string : si la phrase est en minuscule, on la met tout en majuscules, inversement si la phrase est en majuscule, on la transforme en minuscule.

On supposera que tous les strings du tableau sont, soit entièrement minuscules, soit entièrement majuscules pour simplifier cet exercice.

Exemple d'entrée : mapIt(["JE SUIS CONTENTE", 2, " a ", "hello", -23, 4.5])
Sortie attendue : ["je suis contente", "A", "HELLO", 529, 20.25]

Pour travailler avec les majuscules et les minuscules, documentez vous sur les fonctions toLowerCase, toUpperCase.

Exercice 5:

Créez une fonction *cutIt* qui va enlever toutes les occurrences d'une lettre donnée en paramètre d'une phrase, et qui va retourner la phrase obtenue. Pour rappel les Strings sont immuables dans Javascript, vous pouvez changer sa valeur, mais pas l'une de ses lettres.

Exemple d'entrée : cutIt("All the good girl goes to hell", "o")
Sortie attendue : "All the gd girl ges t hell"

Exercice 6:

On veut vérifier si un tableau d'entiers est bel et bien trié dans l'ordre croissant. Pour cela, créez une fonction *isSorted* qui va retourner *true* si le tableau fourni en paramètre est trié dans un ordre croissant et qui va retourner *false* sinon.

Exemple d'entrée : isSorted([1,2,3])
Retour attendue : true
Exemple d'entrée : isSorted([1,3,2,4])
Sortie attendue : false

Exercice 7 :

Créez une fonction qui calcule le PGCD de deux nombres. Une des formules (récursive) possibles pour calculer le PGCD est la suivante :

- pgcd (a, b)= pgcd (a − b, a), si a> b
- pgcd (a, b)= pgcd (a,b − a), si b >a
- pgcd (a, b)= a, si a = b

Exercice 8:

Vous allez créer une fonction *truncate* qui imitera la fonction prédéfinie *slice* de Javascript. Pour info la fonction *slice* donne en sortie une portion du tableau d'origine, la portion est définie par un indice de début et un indice de fin (indice de fin exclu). Votre fonction *truncate* doit prendre en paramètre un indice de début et un indice de fin.

Veillez à émettre un message d'erreur au cas où les indices données sont incorrects : indice de début négatif, indice de fin qui dépasse l'indice maximal du tableau.

Exemple d'entrée : *animaux* = ['fourmi', 'boeuf', 'chameau, 'canard', 'éléphant'];

Résultat attendu :

truncate(animaux, 2, 4) va donner ['chameau', 'canard'].

truncate (animaux, 0,1) va donner ['fourmi']

truncate(animaux, 0,0) va donner []

Exercice 9 :

Contexte :

Jean et Pauline sortent ensemble. Jean habite à 5 km de chez Pauline et malheureusement aucun d'eux n' a de téléphone avec eux. Pourtant ils veulent constamment se parler. Fort heureusement, Ken, le transporteur de lait passe chez Pauline puis chez Jean. Alors, Ken emmène la lettre venant de Pauline pour la donner à Jean. Et inversement. Pour garder toute confidentialité, Pauline et son copain ont décidé de coder les lettres qu'ils s'envoient.

Ils écrivent donc toutes les phrases à l'envers, c'est-à-dire que tous les mots sont écrits dans l'ordre inverse que la normale. Et chaque mot contient un J au début et un P à la fin.

Crée donc une fonction *decode* qui va décoder le message.

Entrée : *decode*(" jenceintep jsuisp jjep jquep jcroisp jjep jJeanp")

Sortie : "Jean je crois que je suis enceinte"

Exercice 10 :

Écrivez une fonction *goOnDateWithJohnny*

Contexte :

Johnny est un gars très beau gosse et très intelligent. Il est aussi très riche. Il est donc le gars très séduisant que toutes les filles s'arrachent. Le seul hic : Johnny est très sélectif et superficiel. Il vous demande alors de filtrer toutes ses prétendantes pour voir qui mérite d'aller en date avec lui. Chaque prétendante sera symbolisée sous forme de tableau dans lequel il y aura les critères suivantes :

- son nom
- son âge
- sa taille en cm
- la couleur de ses yeux
- la couleur de ses cheveux
- sa taille en silhouette (XS, S, L, M, XL, XXL)
- son pays de provenance

Exemple : ["Corine", 23, 170, "bleu", "rousse", "XS", "Angleterre"]

Voici les critères de Johnny :

➔ Entre 20 et 26 ans

➔ Pas moins de 165 cm de longueur et pas plus de 175 cm.

➔ Avec des yeux bleus ou verts

➔ Des cheveux qui ne sont pas rousses

➔ Taille S ou L ou M

➔ Du même pays que lui : Allemagne

Si une prétendante ne respecte pas une de ces critères elle est immédiatement disqualifiée. La fonction *goOnDateWithJohnny* va prendre en paramètre une groupe de fille, c'est-à-dire un tableau de prétendantes, elles-mêmes représentées chacunes par un tableau. Vous aurez donc ce qu'on appelle un tableau à deux dimensions.

La fonction doit retourner un tableau contenant uniquement les prétendantes qui correspondent aux critères de Johnny.

Exemple entrée :

goOnDateWithJohnny (

[

["Sofia", 25, 167, "bleu", "brune", "M", "Allemagne"],

["Jeanne", 27, 169, "vert", "rousse", "S", "Allemagne"],

["Karlota", 23, 166, "noir", "noir", "L", "Italie"],

["Zoe", 23, 165, "bleu", "blonde", "S", "Allemagne"]

]

);

Sortie attendue : [["Sofia", 25, 167, "bleu", "brune", "M", "Allemagne"], ["Zoe", 23, 165, "bleu", "blonde", "S", "Allemagne"]]

Challenge additionnel :

Résolvez deux problèmes d'algorithmique de votre choix sur chacun des sites suivants :

- Codingame (niveau facile ou moyen);
- Codewars (7th kyu ou 6th kyu de préférence);
- Battle dev (niveau 1 et/ou niveau 2)