# Enoncé 1 : int calculCheckDigit (int[] v)

Un numéro de carte de crédit contient 16 chiffres, le dernier chiffre de droite est calculé à partir des 15 premiers chiffres, ceci permet de faire un premier contrôle de validité lorsqu’un utilisateur encore un numéro de carte sur un site Internet.

On va considérer dans l’exercice que le numéro de carte est maintenu dans un vecteur d’entiers de 16 cases où chaque case ne pourra contenir qu’un seul chiffre. Voici comment calculer le check digit :

Exemple d’un numéro dont on désire calculer son dernier chiffre (check digit)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| V | 4 | 5 | 7 | 1 | 5 | 8 | 2 | 3 | 4 | 4 | 8 | 9 | 6 | 3 | 1 | ? |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

Etape 1 : multiplier par 2 les cases d’indice pair et soustraire 9 si la multiplication dépasse 10.

Simulation de la multiplication mais à ne pas faire dans le vecteur :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| V | 8 | 5 | 14 | 1 | 10 | 8 | 4 | 3 | 8 | 4 | 16 | 9 | 12 | 3 | 2 | ? |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

Si la valeur dépasse 10, on lui soustrait 9 ce qui donnera :

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| V | 8 | 5 | 5 | 1 | 1 | 8 | 4 | 3 | 8 | 4 | 7 | 9 | 3 | 3 | 2 | ? |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

Etape 2 : le check digit sera la somme de toutes les cases (à l’exception de la cellule d’indice 15 évidemment) que l’on multiplie par 9 modulo 10 :

Ici la somme fait 71 donc 71 \* 9 = 639 % 10 = 9 🡪 le check digit sera donc un 9 !

Fonction : int calculCheckDigit (int[] v)

Hypothèses de départ :

Le vecteur est un vecteur d’entiers de 16 cases contenant des chiffres (0..9)

Variables d’entrée de la fonction (nom, type et description) :

V : un vecteur d’entiers comme indiqué dans les hypothèses

Variables de sortie de la fonction :

Un entier entre 0 et 9

Variables locales (nom, type, description) :

# Enoncé 2 : int maxValeur (int[] jet)

Réalisez une fonction qui indique un poker ou un carré à partir d’un jet de 5 dès.

Considérons un jet de 5 dès donnant ainsi 5 valeurs allant de 1 à 6 qui seront représentées dans un vecteur.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Jet1 | 1 | 1 | 2 | 1 | 1 |
|  |  |  |  |  |  |
| Jet2 | 1 | 1 | 2 | 2 | 1 |
|  |  |  |  |  |  |
| Jet3 | 1 | 1 | 1 | 1 | 1 |

Fonction int maxValeur (int[] jet)

La fonction doit renvoyer :

* 1 si le vecteur contient 5 valeurs identiques (exemple Jet3 🡺 1)
* 2 si le vecteur contient 4 valeurs identiques (exemple Jet1 🡺 2)
* 0 dans les autres cas (exemple Jet2 🡺 0)

Hypothèses de départ :

La fonction reçoit en entrée un vecteur de 5 cases contenant chacune un chiffre allant de 1 à 6.

Variables d’entrée de la fonction (nom, type et description) :

Jet : vecteur d’entiers de 5 cases

Variables de sortie de la fonction :

Un entier ayant comme valeur 0, 1 ou 2.

Variables locales (nom, type, description) :

# Enoncé 3 : int[] interV (int[] L1, int[] L2)

Soit 2 vecteurs d’entiers triés par ordre croissant. On vous demande d’écrire le pseudocode d’une fonction qui retourne un nouveau vecteur d’entiers trié contenant les valeurs de L1 qui existent dans L2.

Exemple :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| L1 | 1 | 2 | 5 | 6 | 8 |
|  |  |  |  |  |  |
| L2 | 4 | 5 | 6 |  |  |

Le retour de la fonction devra être :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| L3 | 2 | 5 | 6 | 0 |  |

Fonction int[] interV (int[] L1, int[] L2)

Hypothèses de départ :

Les valeurs des vecteurs L1 et L2 sont des entiers.

Le vecteur de sortie L3 aura comme taille min(taille de L1, taille de L2)+1.

L3[0] contiendra le nombre d’éléments de l’intersection.

Variables d’entrée de la fonction (nom, type et description) :

L1 : vecteur d’entiers trié

L2 : vecteur d’entiers trié

Variables de sortie de la fonction :

L3 : vecteur d’entiers trié contenant en première case le nombre d’éléments en commun suivi de la liste des éléments en commun.

Variables locales (nom, type, description) :

# Enoncé 4 : int[] nbPionsColonne (int[][] jeu)

Calculez le nombre de cases occupées pour chaque colonne.

Considérons un jeu de puissance 4 qui comprend 6 lignes et 7 colonnes. On va le représenter dans une matrice de 9 sur 8 pour définir une bordure.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 |
| 2 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 |
| 3 | -1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | -1 |
| 4 | -1 | 0 | 0 | 2 | 1 | 2 | 0 | 0 | -1 |
| 5 | -1 | 0 | 1 | 2 | 2 | 1 | 0 | 0 | -1 |
| 6 | -1 | 2 | 1 | 1 | 1 | 2 | 2 | 0 | -1 |
| 7 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

-1 : bordure

0 : case vide

1 : pion jaune

2 : pion rouge  
Dans ce jeu, un joueur insère un pion dans une colonne non remplie et celui-ci descendra jusqu’à ce qu’il rencontre un autre pion ou le fond de la colonne.

Créez une fonction qui retourne un vecteur de 7 cases reprenant le nombre de cases occupées pour chaque colonne.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 4 | 3 | 3 | 1 | 0 |

Dans l’exemple : res 🡺

Fonction int[] nbPionsColonne (int[][] jeu)

Hypothèses de départ :

La matrice sera une matrice d’entiers de 8 sur 9 encodée comme ci-dessus.

Variables d’entrée de la fonction (nom, type et description) :

Jeu : matrice d’entiers de 8 sur 9.

Variables de sortie de la fonction :

Res : vecteur de 7 cases reprenant des valeurs allant de 0 à 6.

Variables locales (nom, type, description) :

# Enoncé 5 : int[] fusionListe (int[] L1, int[] L2)

Soient 2 listes d’entiers triées par ordre croissant. On vous demande d’écrire le pseudocode d’une fonction qui retourne une nouvelle liste d’entiers triée qui sera la fusion des 2 listes reçues en entrée.

Exemple :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| L1 | 2 | 5 | 6 | 8 |  |
|  |  |  |  |  |  |
| L2 | 4 | 6 |  |  |  |

Le retour de la fonction devra être :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| L3 | 2 | 4 | 5 | 6 | 6 | 8 |

Fonction int[] fusionListe (int[] L1, int[] L2)

Hypothèses de départ :

Variables d’entrée de la fonction (nom, type et description) :

Variables de sortie de la fonction :

Variables locales (nom, type, description) :

# Enoncé 6 : boolean ajoutValeur(int[][] jeu, int chiffre, int x, int y)

Créez une fonction qui valide un coup de Sudoku.

Le jeu de Sudoku consiste à remplir une grille de 9x9 cases avec une série de chiffres (1 à 9) tous différents, qui ne se trouvent jamais plus d’une fois sur une même ligne, dans une même colonne ou dans un même bloc (on distingue 9 blocs de 3x3 dans la grille). On va considérer ici une matrice de 9x9 d’entiers où le 0 représente une case vide.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | 3 | 0 | 0 | 8 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 1 | 9 | 5 | 0 | 0 | 0 |
| 0 | 9 | 8 | 0 | 0 | 0 | 0 | 6 | 0 |
| 8 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 3 |
| 4 | 0 | 0 | 8 | 0 | 3 | 0 | 0 | 1 |
| 7 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 6 |
| 0 | 6 | 0 | 0 | 0 | 0 | 2 | 8 | 0 |
| 0 | 0 | 0 | 4 | 1 | 9 | 0 | 0 | 5 |
| 0 | 0 | 0 | 0 | 8 | 0 | 0 | 7 | 9 |

On vous demande d’écrire une fonction qui reçoit en paramètre la grille 9x9, le chiffre (1 à 9) à rajouter et la position (x, y) où l’on désire mettre la valeur.

Votre fonction devra :

* Si le coup est valable : (case vide et pas de conflit)
  + Renvoyer « true »
  + Mettre à jour la grille en rajoutant le chiffre
* Si le coup n’est pas valable : case occupée ou au moins un conflit)
  + Renvoyer « false »

Fonction boolean ajoutValeur(int[][] jeu, int chiffre, int x, int y)

Hypothèses de départ :

Jeu : matrice d’entiers de 9x9 avec des 0 dans les cases vides. On suppose que cette matrice est correcte, c’est-à-dire qu’elle ne possède pas de conflit.

Chiffre :

X, y :

Variables d’entrée de la fonction (nom, type et description) :

Jeu : matrice d’entiers de 9x9

Chiffre : entier entre 1 et 9, valeur qu’on veut encoder

X, y : entier, position de la case

Variables de sortie de la fonction :

Booléen qui donnera vrai si le coup est valable et ajuste la matrice, faux si le coup est invalide.

Variables locales (nom, type, description) :

# Enoncé 7 : int estUneSuite(int[] jet)

Réalisez une fonction qui analyse le jet de 5 dès. Considérons un jet de 5 dès donnant ainsi 5 valeurs allant de 1 à 6 qui seront représentées dans un vecteur.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Exemple 1 : | Jet | 1 | 2 | 5 | 4 | 3 | * Petite suite |
|  |  |  |  |  |  |  |  |
| Exemple 2 : | Jet | 4 | 6 | 2 | 5 | 3 | * Grande suite |

Fonction int estUneSuite(int[] jet)

La fonction doit renvoyer :

* 1 si le vecteur contient une petite suite (avec les chiffres de 1 à 5 pas nécessairement dans l’ordre)
* 2 si le vecteur contient une grande suite (avec les chiffres de 2 à 6 pas nécessairement dans l’ordre)
* 0 dans les autres cas

Hypothèses de départ :

La fonction reçoit en entrée un vecteur de 5 cases contenant chacune un chiffre allant de 1 à 6.

Variables d’entrée de la fonction (nom, type et description) :

Jet : vecteur d’entiers de 5 cases.

Variables de sortie de la fonction :

Un entier ayant comme valeur 0, 1 ou 2.

Variables locales (nom, type, description) :