intech
ECOLE SUPERIEURE
D'INGENIERIE INFORMATIQUE

EXERCICE : UNICORN

invenietis

1 Énoncé

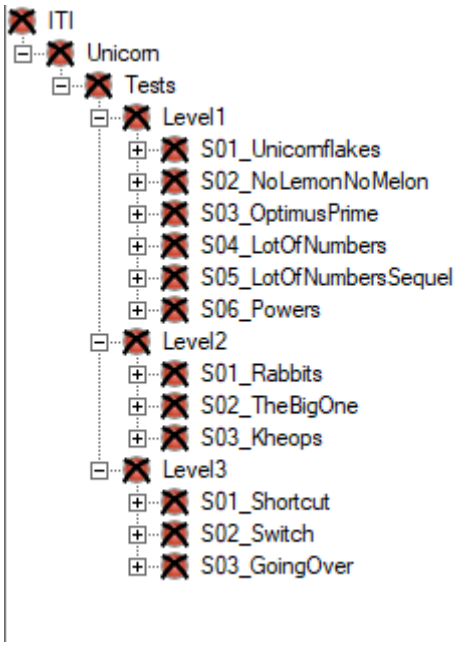
L'objectif de cet exercice est de résoudre un ensemble de problèmes mathématiques et algorithmiques simples.

Vous disposez pour cela d'une solution comprenant 2 projets :

- *ITI.Unicorn.Tests*, contiens les tests unitaires
- *ITI.Unicorn.Core*, contient les méthodes à implémenter

Pour faire tourner les tests unitaires, il vous suffit de configurer le projet *ITI.Unicorn.Tests* en tant que projet de démarrage puis d'exécuter la solution.

Comme vous pouvez le constater, pour le moment, tous les tests sont rouges :



À vous de faire en sorte qu'ils passent en vert. Pour cela, vous avez le droit de faire ce que bon vous semble dans le projet *ITI.Unicorn.Core*.

Les tests unitaires sont là pour spécifier de façon détaillée les fonctionnalités attendues. Cependant le présent document va décrire brièvement ce qui est attendu pour les différents problèmes.

2 Implémentation des différents problèmes

Certains tests ont un attribut *Timeout* défini, indiquant un temps maximal autorisé en millisecondes. Les limites de temps définies sont généreuses. Si votre test est rouge pour cause de timeout, revoyez votre implémentation.

2.1 Level 1

2.1.1 Unicornflakes

Le but de l'exercice est de transformer un tableau de Int en un tableau de String en ayant les multiples de 3, 5 et 7 remplacés par des chaines de caractères définies selon les critères suivants :

- Les multiples de 3 : uni
- Les multiples de 5 : corn
- Les multiples de 7 : flakes
- Les multiples de 3 et 5 : unicorn
- Les multiples de 3 et 7 : uniflakes
- Les multiples de 5 et 7 : cornflakes
- Les multiples de 3 et 5 et 7 : unicornflakes

Écrire une fonction qui prend en argument un tableau d'entiers et retourne un tableau de string ayant subi les transformations ci-dessus.

2.1.2 No lemon, no melon

On cherche à trouver le plus grand palindrome produit par des facteurs ayant le même nombre de chiffres.

Écrire une fonction qui prend en argument le nombre de chiffres des facteurs et retourne le plus grand palindrome.

Par exemple, si on passe 2 à la fonction, elle doit retourner le plus grand palindrome qui est le produit de deux nombres à 2 chiffres, soit 9009.

2.1.3 Optimus Prime

En parcourant le début de la liste des nombres premiers : 2, 3, 5, 7, 11, et 13, on peut voir que le 5ème nombre premier est le 11.

Écrire une fonction qui prend en argument un nombre n et retourne le $n^{ième}$ nombre premier.

2.1.4 Lot of numbers

Écrire une fonction qui prend en argument une string de chiffres et un nombre n. Cette fonction retourne le plus grand produit de n chiffre consécutif dans la string.

2.1.5 Lot of numbers Sequel

Écrire une fonction qui prend en argument un tableau à 2 dimensions. Cette fonction retourne le plus grand produit de 4 nombres consécutifs contenus dans le tableau, que ce soit vers le bas, le haut, la droite, la gauche ou diagonalement.

2.1.6 Powers

Seulement trois nombres peuvent être écrits comme la somme des puissances de 4 de leurs chiffres :

$1634 = 1^4 + 6^4 + 3^4 + 4^4$
$8208 = 8^4 + 2^4 + 0^4 + 8^4$
$9474 = 9^4 + 4^4 + 7^4 + 4^4$

Écrire une fonction qui prend en argument une puissance p et qui retourne la somme de tous les nombres qui peuvent s'écrire comme la somme des puissances p de leurs chiffres.

En reprenant l'exemple précédent, où p = 4, le résultat serait 1634 + 8208 + 9474 soit 19316.

2.1.7 Wall Street

Écrire une fonction qui prend en entrée un tableau d'entiers représentant l'évolution journalière des prix du bitcoin et retourne le profit maximal pour une transaction (1 achat, 1 vente).

Par exemple, avec les prix suivants :

[2, 1, 5, 3, 6, 4]

Le profit maximal est 5 (achat lorsque le bitcoin vaut 1 et vente quand il vaut 6).

2.2 Level 2

2.2.1 Rabbits

Écrire une fonction qui prends en argument un nombre n et retourne la somme de tous les nombres pairs de la suite de Fibonacci inférieur à n.

Rappel : *La valeur d'un INT est compris entre -2,147,483,648 et 2,147,483,647, attention à l'overflow.*

2.2.2 The big one

On définit une suite itérative par :

- n -> n /2 si n est pair
- n -> 3n +1 si n est impair

On s'arrête une fois que le nombre 1 est atteint.

Par exemple, si on prend 5 comme terme initial, on obtient alors une séquence de 6 nombres :

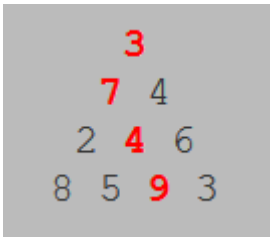
5 → 16 → 8 → 4 → 2 → 1

Selon les règles définies précédemment, écrire une fonction qui prend en argument un nombre n et retourne le terme initial compris entre 1 et n qui génère la plus grande séquence.

2.2.3 Khéops

On a une pyramide de nombres. Trouver le chemin reliant le sommet à la base qui, lorsqu'on fait la somme des nombres rencontrés, retourne le plus grand résultat.

Une ligne ne peut être visitée qu'une seule fois.



Ainsi, pour cette pyramide, le chemin le plus gros est 3 + 7 + 4 + 9 soit 23.

Écrire une fonction qui prend en argument la pyramide (un tableau de tableau) et retourne la valeur du chemin produisant la plus grosse somme.

2.2.4 Forest

On considère une forêt rectangulaire divisée en carrés. Chaque carré contient un certain nombre d'arbres. La forêt est représentée par un tableau à 2 dimensions, les valeurs du tableau indiquant le nombre d'arbres dans chaque carré.

Un bucheron arrive dans la forêt et coupe des arbres, carré par carré, en commençant au centre de la forêt. Si celui-ci ne correspond à aucun carré (si la forêt compte un nombre pair de carrés en longueur ou largeur), le bucheron commence dans le carré le plus proche du centre qui contient le plus d'arbres.

Par exemple, dans la forêt suivante, le carré de départ est le 7 (en gras) :

0 2 3 4
6 4 5 2
9 **7** 0 1
3 2 4 5

Une fois que le bucheron a coupé les arbres de son carré, il se déplace vers le haut, le bas, la droite ou la gauche (pas de diagonales) dans le carré adjacent comptant le plus d'arbres non coupés. Si aucun des 4 carrés où il pourrait de déplacer ne contient d'arbres non coupés, il s'arrête.

Écrire une fonction qui prend en entrée une forêt et retourne le nombre d'arbres coupés.

NB1 : si plusieurs carrés possibles pour se déplacer contiennent le même nombre d'arbres, le bucheron choisira le premier dans l'ordre suivant : haut, bas, gauche, droite.

NB2 : l'axe vertical de la forêt est représenté par le deuxième coordonnée dans le tableau à 2 dimensions (ce qui ne correspond pas à la représentation visuelle du tableau dans le code).

Par exemple, avec la forêt précédente, le chemin serait :

0 2 3 4
6 4 5 2
9 **7** 0 1
3 2 4 5

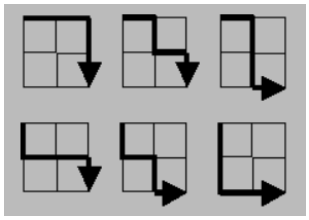
7 → 9 → 6 → 4 → 5 → 3 → 4 → 2 → 1 → 5 → 4 → 2 → 3

D'où le résultat : 55

2.3 Level 3

2.3.1 Shortcut

Dans une grille de 2x2 où l'on peut se déplacer uniquement vers le bas et la droite, il existe 6 chemins différents pour arriver en bas à droite (en partant d'en haut à gauche).



Écrire une fonction qui prend en argument deux nombres *a* et *b* correspondant à la largeur et la hauteur du rectangle. Cette fonction retourne le nombre de chemins possible pour traverser la grille.
Il existe au moins deux solutions différentes. Avec et sans boucle

2.3.2 Switch

Une permutation est un arrangement ordonné de nombres distincts. La liste des permutations des nombres {0, 1, 2} triée dans l'ordre croissant est :

012 021 102 120 201 210

Écrire une fonction qui prend en argument un tableau de nombre et un nombre n. Cette fonction retourne la $n^{ième}$ permutation, par ordre croissant.

En reprenant l'exemple précédent, la 3ème permutation est 102.

Rappel : le nombre de permutations d'un tableau à n éléments est n!

2.3.3 Going Over

Écrire une fonction qui prend en entrée un tableau *array* et retourne en sortie le nombre de paires (i, j) telles que **i < j** et **array[i] > 2*array[j]**.

Par exemple, avec le tableau suivant :

[1, 3, 2, 3, 1]

Le résultat serait 2 (les paires sont (1,4) et (3,4)).

À VOUS DE JOUER !