TP4 : Construisons des murs.

Préliminaires.
Téléchargez depuis le site l'archive `TP4.zip` et extrayez les fichiers dans votre dossier personnel puis vérifiez la compilation du fichier `TP4.cpp`.

Les évènements politiques récents remettent au goût du jour la **construction de murs**. Pour rester en phase avec notre époque, le but du TP de cette semaine est de fabriquer **tous les murs possible** selon certaines contraintes.

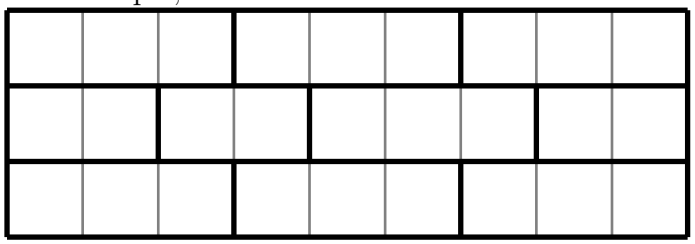
Nous utilisons la définition suivante :

(1) Une ligne de brique de longueur k est composée de briques de **taille 2** ou de **taille 3** telle que la somme des tailles de briques soit k.

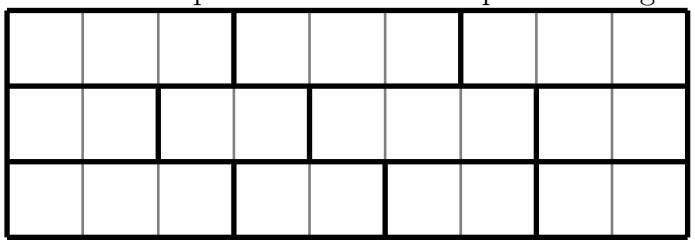
(2) On peut superposer 2 lignes de briques de même taille si **les jointures entre les briques** ne sont pas au même endroit.

(3) Un mur de hauteur n et de largeur k est une superposition de n lignes de longueur k (pour chaque ligne, on doit comparer ses jointures avec la ligne précédente).

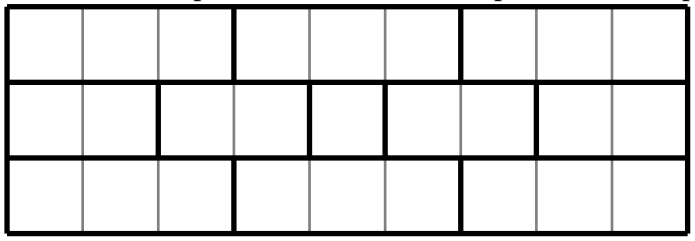
Par exemple, ceci est un mur de hauteur 3 et de longueur 9.



Celui-ci n'est pas un mur car les 2 premières lignes ne sont pas compatibles.

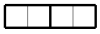
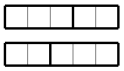
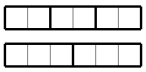
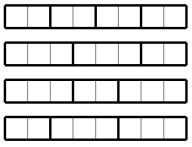


Celui-ci n'est pas un mur car il comporte une brique de taille 1.



Exercice 1 (Comptons les lignes de briques)**.**
En suivant des méthodes similaires à la séance précédente, nous commençons par compter les lignes de briques (les murs de hauteur 1).

Voici la liste des lignes possibles pour les tailles de 0 à 8.

Taille	Nombre	Lignes
0	1	(le mur vide)
1	0	(car pas de briques de taille 1)
2	1	
3	1	
4	1	
5	2	
6	2	
7	3	
8	4	

Je peux compter les lignes de taille 9 de cette façon : 3 lignes de taille 7 auxquelles je rajoute une brique de taille 2 (à la fin) + 2 lignes de taille 6 auxquelles je rajoute une brique de taille 3. c'est-à-dire $3 + 2 = 5$.

(1) Listez les 5 lignes de taille 9 en séparant bien celles qui terminent par une brique de taille 2 et celles qui terminent par une brique de taille 3.

(2) Combien y-a-t-il de lignes de taille 10 ? 11 ? 12 ? (Suivez la même logique que précédemment)

(3) Comment exprimer le nombre de lignes de taille n, $f(n)$ en fonction des valeurs précédentes de la suite ?

(4) Implantez les fonctions `nbLignesRec` et `nbLignesDyn` selon les méthodes récursives et dynamiques vues lors de la séance précédente et lancez les tests correspondants.

Exercice 2 (Premières lignes)**.**
Dans la suite de ce TP, nous allons chercher à *engendrer* toutes les lignes et tous les murs possible. Pour cela, on va représenter les lignes sous forme de *tableau de booléens*. Les booléens représentent les *intersections* de blocs de taille 1 : si une intersection est entre 2 briques, on met `true`, sinon on met `false`.

Observez par exemple la ligne suivante de taille 8 : elle est représentée par un tableau de taille 9, il y a `true` si on est sur un trait épais (entre deux briques) et `false` sinon.

true false false true false true false false true

Pour représenter une ligne de taille n, un tableau de taille $n + 1$ doit
— commencer et terminer par `true`,
— ne jamais avoir deux valeurs `true` d'affilée,
— ne jamais avoir plus de 2 valeurs `false` d'affilée.

(1) Regardez la variable `exempleLigne2` définie dans le fichier et dessinez la ligne correspondante à la main puis affichez cette ligne avec la fonction `afficheLigne` qui vous est donnée.

(2) Pourquoi les 4 tableaux `contreExemple1`, `contreExemple2`, `contreExemple3`, `contreExemple4` ne représentent-ils pas des lignes ?

(3) Implantez la fonction `estLigne` qui teste qu'un tableau de booléen représente bien une ligne et lancez les tests correspondants.

(4) Implantez la fonction `formatLigneMur` qui vérifie qu'un mur (tableau de lignes) est composé de lignes valides et de même tailles. Lancez les tests correspondants.

(5) On cherche maintenant à savoir si deux lignes sont compatibles pour être empilées dans un mur : pour cela, il faut que les jointures de briques se situent à des positions différentes. Implantez la fonction `jointuresDistinctes` et lancez les tests correspondants.

(6) Implantez la fonction `jointuresLignesMur` qui vérifie que les lignes du mur sont compatibles en terme de jointures.

Exercice 3 (Fabriquons des lignes de briques)**.**
On vous donne une fonction `genereBooleens` qui fabrique l'ensemble de tous les tableaux de booléens possible de taille k. Cette fonction suit une structure très similaire à celles que nous avons vue lors de la séance précédente.

(1) **Utilisez** les fonctions `genereBooleens` et `estLigne` pour fabriquer l'ensemble des lignes de briques de taille k dans la fonction `genereLignesMethode1`. Le principe est le suivant : on fabrique **tous** les tableaux de booléens et on ne garde que ceux qui sont bien des lignes.

(2) Cette méthode est très peu efficace car la plupart des tableaux de booleéns ne sont pas des lignes, on en fabrique donc beaucoup trop. On propose donc d'implanter une seconde méthode selon ce principe : pour fabriquer les lignes de taille k, je fabrique toutes les lignes de taille $k-2$ et je rajoute une brique de taille 2 puis je fabrique toutes les lignes de taille $k-3$ et je rajoute une brique de taille 3. La structure de la fonction vous est donnée dans `genereLignesMethode2`, à vous de compléter. Les fonctions `ajouteBrique2` et `ajouteBrique3` vous sont données et vous pouvez les utiliser.

Exercice 4 (Fabriquons des murs)**.**
Un mur est simplement un tableau de lignes compatibles (chaque ligne est compatible avec la précédente). Le but de l'exercice est de fabriquer tous les murs d'une hauteur et largeur donnée.

(1) ♣ En s'inspirant des méthodes vues précédemment, implantez la fonction `genereMurs` qui retourne l'ensemble des murs de hauteur n et de largeur k. Votre fonction sera *récursive* car pour générer les murs de hauteurs n, vous utiliserez les murs de hauteurs $n-1$. Par ailleurs, il faut utiliser la fonction `genereLignes` (méthode 1 ou 2).

(2) ♣♣ On souhaite améliorer la méthode précédente pour ne pas fabriquer trop de lignes inutiles. Pour cela, on décide d'implanter une fonction `genereLignesMethode3` qui fabrique l'ensemble des lignes compatibles avec une lignée donnée. Puis on utilise cette fonction pour écrire `genereMursMethode2`. Implantez ces deux fonctions.