exercice le programme de l'exercice 3.5, qui résout les tours de Hanoï, en un programme non récursif.

Pour déplacer un paquet formé de `k` disques de la colonne `x` vers la colonne `y`, on commence par déplacer `k - 1` disques de la colonne `x` vers la colonne `m`, puis `1` disque de la colonne `x` vers la colonne `y` et finalement `k - 1` disques de la colonne `m` vers la colonne `y`, où `m` est l'unique colonne distincte de `x` et `y`. Cela peut se programmer de manière récursive

```
static void h (final int x, final int y, final int k) {
 if (k == 1) System.out.print(x + " -> " + y + "  ");
 else {int m = 6 - (x + y);
       h(x,m,k - 1);
       h(x,y,1);
       h(m,y,k - 1);}}
```

Une alternative à l'utilisation de la récursivité est l'utilisation d'une pile qui contient l'ensemble des mouvements de paquets en attente d'être exécutés. Chaque mouvement se décrit par un triplet d'entiers formé de l'origine du paquet à déplacer, de sa destination et de sa taille.

Tant que la pile n'est pas vide, on examine son sommet. S'il décrit le mouvement d'un paquet formé d'un unique disque, on affiche le mouvement de ce disque et on supprime le sommet. S'il contient un mouvement d'un paquet formé de `k` disques, on le remplace par trois descriptions de mouvements de paquets plus petits.

Écrire un programme qui résout les tours de Hanoï sans utiliser la récursivité.

Exercice 6.14

On considère une liste de caractères dans l'ensemble `'('`, `')'`, `'['`, `']'`. L'ensemble des listes bien formées est le plus petit ensemble tel que

- la liste vide est bien formée,
- si `l` est bien formée alors `(l)` et `[l]` également,
- si $\texttt{l}_1$ et $\texttt{l}_2$ sont bien formées alors $\texttt{l}_1\ \texttt{l}_2$ également.

Par exemple, la liste `([[()]()][])` est bien formée, mais pas la liste `([[()]()]][])`.

Écrire une fonction qui indique si une liste de caractères est bien formée ou non.

Pour une fois, il est plus simple d'utiliser une pile que la récursivité.