INF4100
Laboratoire no. 4

06/02/08

(Diviser-pour-régner et équations de récurrence)

1. Fonction `palindrome` et équations de récurrence

On appelle *palindrome* un mot ou texte qui reste le même qu'on le lise de gauche à droite ou de droite à gauche, par exemple, «laval», «kayak».

a. Écrivez un programme `palindromes.mpd` qui reçoit en argument (au moment de l'appel sur la ligne de commande) une chaîne de caractères et qui détermine si cette chaîne est un palindrome. Ceci doit se faire à l'aide d'une fonction `estPalindrome` qui utilise une approche récursive (stratégie diviser-pour-régner).

   Dans cette première version, vous pouvez traiter les espaces blancs comme n'importe quel autre caractère, i.e., il faut aussi que les blancs «*matchent*» correctement ensemble pour que la chaîne soit considérée comme un palindrome, par exemple :

```
% a.out "kayak"
estPalindrome( "kayak" ) => true
% a.out "kayak "
estPalindrome( "kayak " ) => false
% a.out " la v al "
estPalindrome( " la v al " ) => true
% a.out " la val "
estPalindrome( " la val " ) => false
```

   Sélectionnez une opération barométrique appropriée puis donnez les équations récurrence décrivant le nombre d'opérations barométriques exécutées par votre fonction récursive en fonction de n (taille de la chaîne). Déduisez en la complexité asymptotique de votre fonction.

b. Pour déterminer si un texte formé de nombreux mots est un palindrome, on ignore généralement les espaces blancs, accents, minuscules/MAJUSCULES, caractères de ponctuation, etc., par exemple :[1]

   - «Ésope reste ici et se repose»
   - «Un art luxueux ultra nu»
   - «Et la marine va venir à Malte»

   Modifiez votre programme de façon à déterminer si une chaîne est un palindrome, mais en ignorant les espaces blancs et les différences entre minuscules/MAJUSCULES — vous pouvez supposer que les caractères alphabétiques n'ont pas d'accent.

Quelques éléments du langage MPD, utiles pour cet exercice, sont présentés à la page 3.

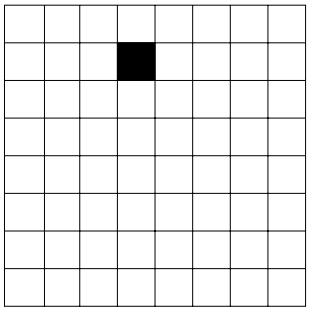
[1] De nombreux autres exemples sont présentés sur le site Web `http://villemin.gerard.free.fr/Langue/Palindro.htm`. Un palindrome célèbre (en français) est celui de Georges Perec (1969), qui contient 6372 caractères (1247 mots) : `http://worldserver2.oleane.com/fatrazie/GdPalindrome.htm`. Perec est aussi l'auteur de «La disparition», un roman de 319 pages écrit... sans utiliser une seule fois la lettre 'e'!

2. Couverture de *trominos* et équations de récurrence

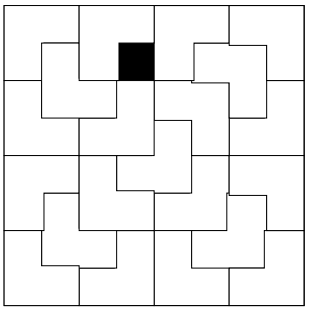
Supposons que l'on ait un échiquier de taille $n \times n$ — avec n une puissance de 2 — où l'une des cases a été «retirée» — par exemple, on a fait un trou : voir Figure 1 (a).

Un *tromino* est une pièce «en équerre» (en L) pouvant couvrir trois cases de l'échiquier.

Soit alors un ensemble de trominos, numérotés de 1 à $\frac{n^2-1}{3}$. On veut utiliser ces trominos pour couvrir l'ensemble de l'échiquier, mais en laissant libre la case retirée (trou), tel qu'illustré à la Figure 1 (b).



(a) Échiquier initial



(b) Échiquier couvert

```
{Devoir1@linux} trominos 8 2 4 1
a::
 1  1  2  2  6  6  7  7
 1  5  2  0  6 10 10  7
 3  5  5  4  8  8 10  9
 3  3  4  4 21  8  9  9
11 11 12 21 21 16 17 17
11 15 12 12 16 16 20 17
13 15 15 14 18 20 20 19
13 13 14 14 18 18 19 19
```

(c) Résultat du programme

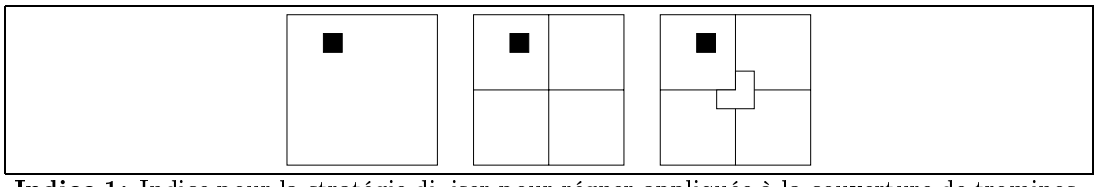
Figure 1: Un échiquier 8×8 avec un trou, puis couvert avec des trominos.

a. *Complétez* le squelette de programme fourni sur le site *Web* du cours. Plus précisément, vous devez compléter la procédure `couvrir`, procédure récursive appelée par la procédure `couvrirDeTrominos` — si nécessaire (?), vous pouvez introduire des procédures auxiliaires.

   La procédure `couvrir` doit utiliser une stratégie *récursive* diviser-pour-régner — pour un petit coup de pouce sur comment procéder, voir l'indice 1.

   Un exemple de résultat produit par le programme est présenté à la Figure 1 (c), où la valeur 0 indique la présence du «trou» (2ième ligne, 4ième colonne).

b. Sélectionnez une opération barométrique appropriée puis donnez les équations récurrence décrivant le nombre d'opérations barométriques exécutées par votre fonction récursive en fonction de n (nombre de lignes = nombre de colonnes de l'échiquier). Déduisez en la complexité asymptotique de votre fonction.



Indice 1: Indice pour la stratégie diviser-pour-régner appliquée à la couverture de trominos.

- Une chaîne peut être déclarée comme suit, où `N` indique la taille maximale de la chaîne :

```
const int N = 1024;
string[N] s;
```

  La variable `s` peut ensuite être utilisée pour recevoir la chaîne fournie en argument du programme :

```
getarg(1, s);
```

  La chaîne fournie n'a pas besoin d'avoir exactement `N` caractères : on peut utiliser l'opération `length(s)` pour déterminer sa longueur effective.

- Une chaîne reçue en argument peut être déclarée à l'aide d'une borne «*», par exemple :

```
procedure f( string[*] s ) returns int r.
```

- Le format «`%s`» peut être utilisé dans un `printf` pour imprimer une chaîne de caractères.
- Le format «`%b`» peut être utilisé dans un `printf` pour imprimer une valeur booléenne.
- La fonction suivante permet de convertir une lettre Majuscule en une lettre minuscule :

```
procedure toLower( char c ) returns char r
{
  r = c;
  if ('A' <= c & c <= 'Z') {
    r = char( int(c) + int('a') - int('A') )
  }
}
```

- Pour plus de détails sur les opérateurs prédéfinis en MPD, vous pouvez consulter l'URL suivant : `http://www.cs.arizona.edu/mpd/SRbook.appendixC.html`.

Indice 2: Quelques éléments utiles du langage MPD pour l'exercice 1.