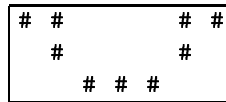


De belles images

Le but de ce TP est de créer un paquetage destiné à la manipulation d'images en noir et blanc : une image est une grille de pixels, allumés ou éteints. Pour représenter une telle image, on choisit d'indiquer les coordonnées des pixels allumés. Par exemple, l'image



est décrite par la liste de coordonnées (1, 1), (1, 2), (1, 6), (1, 7), (2, 2), (2, 6), (3, 3), (3, 4), (3, 5). Il est également utile de disposer des dimensions de l'image. On définit donc un type `Image` composé des informations suivantes :

- les dimensions de l'image,
- les coordonnées des pixels actifs, stockées dans une liste chaînée. On supposera que les coordonnées sont énumérées ligne par ligne et, pour chaque ligne, colonne par colonne.

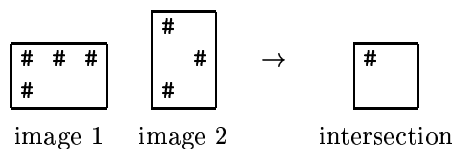
Pour le TP vous sont fournis les squelettes de `paq_image.ads` et `paq_image.adb`, ainsi que deux fichiers `six` et `traits` contenant des exemples d'image et qui pourront vous servir à tester vos fonctions. Tout est dans le répertoire `~/touzet/ADA/Image`.

Question 1. Compléter la spécification du paquetage.

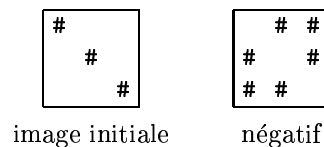
Question 2. La procédure `get(S:in String; I:out Image)` doit permettre de lire une image contenue en clair dans le fichier texte de nom `S` et de la stocker dans `I`. Compléter le code qui vous est proposé pour `get`.

Question 3. Écrire une procédure `put` qui affiche un objet de type `Image` en clair à l'écran.

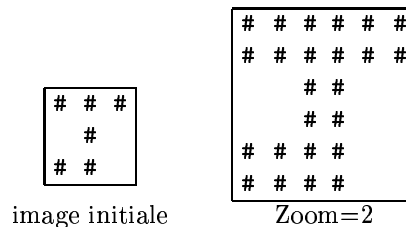
Question 4. Écrire une fonction `Image_commune` qui prend en argument deux images et qui construit leur intersection.



Question 5. Écrire une procédure `Negatif` qui transforme une image en son négatif.



Question 6. Écrire une procédure `Zoom` qui prend en argument une image, un entier positif n et qui transforme l'image en lui appliquant un facteur de grossissement égal à n . Pour cela, on procède grossièrement: chaque pixel est remplacé par n^2 pixels.



Pour cette question, il est recommandé d'utiliser un tableau intermédiaire pour manipuler l'image. Réfléchissez bien.