Exercice 2 (Affichage (1 pt))
Écrivez une procedure qui affiche une grotte sur la sortie standard.

Exercice 3 (Choir ou pas choir (2 pts))
Écrivez une fonction qui test si la goutte peut encore descendre et si oui, qui retourne les 1,2 ou 3 coordonnées possibles où la goutte peut passer : (1) juste en dessous ; (2) si pas possible en dessous, un trou possible vers la gauche ; (3) ou un trou possible vers la droite. Chuter est prioritaire sur avoir un trou vers la gouche ou la droite. Mais il n'y a pas a priori de priorité entre la gauche et la droite.

Exercice 4 (Fin de la simulation (1 pt))
Écrivez une fonction qui test s'il est encore possible ou non de mettre des gouttes. On suppose une grotte pleine quand toute la matrice est pleine.

Exercice 5 (Le "main" (1 pt))
Écrivez la procédure de main qui affiche le déroulement du remplissage de la grotte en choisissant aléatoirement le sommet de la goutte.

Nous entrons dans les 2 questions les plus difficile. Comme la goutte es magique, elle doit choisir le chemin le plus court quand elle rentre dans la grotte. Comme expliqué ci-dessus, la goutte tout en droit tombe en priorité. C'est à chaque fois qu'elle arrive sur un obstacle qu'un choix entre la gauche et la droite doit être réalisé.

Pour ce faire, nous allons construire un arbre (binaire) des chemins possibles. Le noeud correspond aux coordonnées de l'obstacle. Les 2 fils, correspond aux 2 choix possibles (gauche ou droite). Un fils est "null" si un des choix n'est pas possible (il n'existe pas un tel chemin). Ainsi, au final, quand l'arbre sera terminé, on pourra calculer le chemin le plus court. On pourra alors extraire les coordonnées correspondant au chemin que doit parcourir la goutte. La dernière coordonnées sera donc où la goutte doit atterir.

Exercice 6 (ABR des chemins (3 pts))
Écrivez une fonction qui construit un tel arbre.

Exercice 7 (ABR des chemins (2 pts))
Écrivez une fonction qui à partir d'un tel arbre, retourne la coordonnée où doit arriver la goutte.

4 Rappels

Rappel. Un tableau bidimensionnel se déclare, s'initialise et se manipule en ADA :

```
type matrice is array(1..2,1..2) of natural;
M:matrice:=((3,4),(5,6));
.....(* plus loin *).....
M(1,2):=M(2,1)+9;
```

Pour avoir des entiers aléatoires

```
with Ada.Numerics.Discrete_Random;
with Ada.Text_IO;
procedure Roulette is
    type Numero is range 0 .. 36;
    package La_Roulette is new Ada.Numerics.Discrete_Random (Numero);
    use La_Roulette; -- Rend Generator, Reset et Random visibles
    A : Numero;
    G : Generator;
begin
    Reset (G); -- Initialise le générateur (à faire une seule fois)
    A := Random (G); -- Tire un nombre au hasard entre 0 et 36
    Ada.Text_IO.Put_Line ("Le␣numéro␣est:␣" & Numero'Image (A));
end Roulette;
```