On veut représenter un labyrinthe, ainsi que déterminer si un chemin donné amène à la sortie. On le représente par un tableau carré de taille n:

- les cases seront des 0 si l'on peut s'y déplacer,
- les cases seront des 1 s'il s'agit d'un mur.

Exemple:

```
laby=[[0,1,1,1,1,1,1,1,1,1],
      [0,0,0,0,0,0,0,1,0,1],
      [1,0,1,1,1,1,0,1,0,1],
      [1,0,1,0,0,0,0,0,0,1],
      [1,0,1,1,1,1,1,0,1,1],
      [1,0,1,0,0,0,1,0,1,1],
      [1,0,1,0,1,0,1,0,1,1],
      [1,0,1,1,1,0,1,0,1,1],
      [1,0,0,0,0,0,1,0,0,1],
      [1,1,1,1,1,1,1,1,0,0]]
```

L'entrée du labyrinthe se situe à la première case du tableau en haut à gauche et la sortie du labyrinthe se trouve à la dernière case en bas à droite.

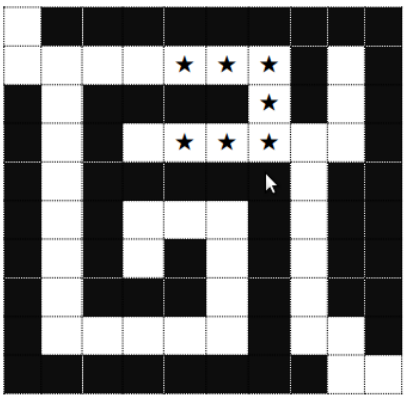
1. Proposer, en langage Python, une fonction `mur()`, prenant en paramètre un tableau représentant un labyrinthe et deux entiers i et j compris entre 0 et n- 1 et qui renvoie un booléen indiquant la présence ou non d'un mur. Par exemple :

```
>>mur(laby, 2, 3)
True
>>mur(laby, 1, 8)
False
```

2. Un parcours dans le labyrinthe va être représenté par une liste de cases. Il s'agit de couples (i,j) où i et j correspondent respectivement aux numéros de ligne et de colonne des cases successivement visitées au long du parcours.

   Exemple:

   La liste `[(0,0),(1,0),(1,1),(5,1),(6,1)]` ne peut correspondre au parcours d'un labyrinthe car toutes les cases parcourues successivement ne sont pas adjacentes.



   Mais la liste `[(1,4),(1,5),(1,6),(2,6),(3,6),(3,5),(3,4)]` correspond au parcours repéré par des étoiles ⋆ ci-contre.

3. Ecrivez une fonction `est_voisine(case1, case2)` prenant deux cases de type *tuple* et renvoyant un booléen indiquant si ces cases sont voisines.

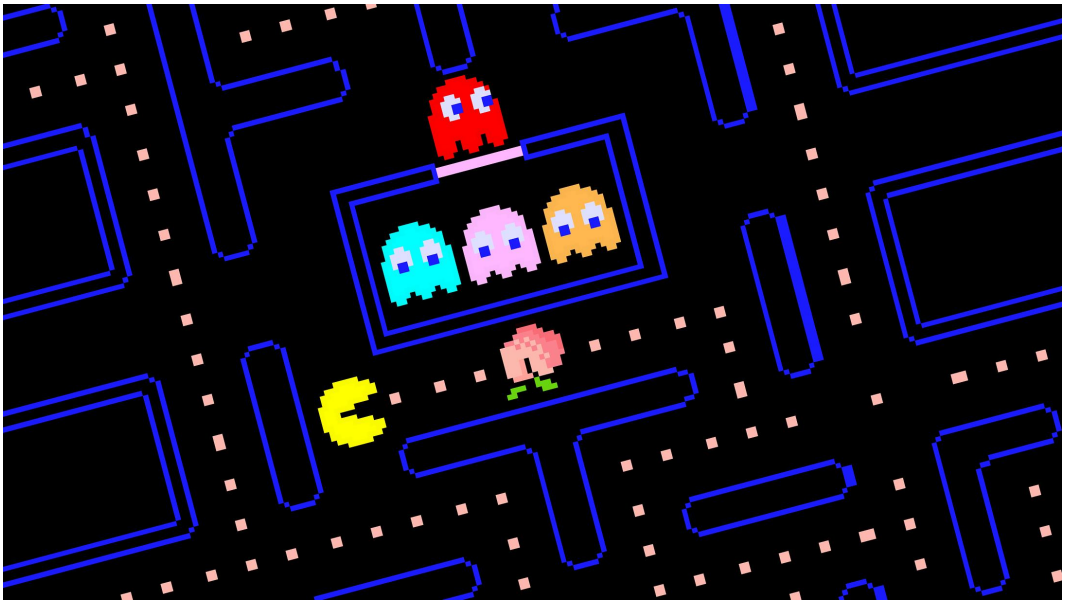


Figure 1: Labyrinthe rétro

4. **Autre option:** On considère la fonction `voisine` ci-dessous, écrite en langage Python, qui prend en paramètres deux cases données sous forme de couple.

```
def voisine(case1, case2) :
    l1, c1 = case1
    l2, c2 = case2
    d = (l1-l2)**2 + (c1-c2)**2
    return (d == 1)
```

   Après avoir remarqué que les quantités `l1-l2` et `c1-c2` sont des entiers, expliquer pourquoi la fonction `voisine` indique si deux cases données sous forme de tuples `(l,c)` sont adjacentes.

5. En déduire une fonction `adjacentes` qui reçoit une liste de cases et renvoie un booléen indiquant si la liste des cases forme une chaîne de cases adjacentes.

   Un parcours sera qualifié de compatible avec le labyrinthe lorsqu'il s'agit d'une succession de cases adjacentes accessibles et non murées.

   Ecrire une fonction `teste(cases, tab)` qui indique si le chemin `cases` est un chemin possible compatible avec le labyrinthe `tab`.

6. En déduire une fonction `echappe(cases, tab)` qui indique par un booléen si le chemin `cases` permet d'aller de l'entrée à la sortie du labyrinthe `tab`.