Correction

NSI - 2021 Étranger Jour 2 (21-NSIJ2G11)

Exercice 2 - Labyrinthe

1.

Solution 1

```
def mur(laby, i, j):
    if laby[i][j] == 1:
        return True
    else:
        return False
```

Solution 2

```
def mur(laby, i, j):
    return laby[i][j] == 1
```

2. a. Pour que deux cases soit adjacente, il faut que la distance qui les sépare vaille 1. La distance est ici calculée selon Pythagore $c = \sqrt{a^2 + b^2}$. La racine carré est omise car $\sqrt{1} = 1$ est le seul cas qui nous intéresse.

2. b.

```
def adjacentes(liste):
    """ Vérifie que la liste est une suite de cases voisines."""
    if len(liste) <= 1:
        return True
    for i in range(1, len(liste)):
        if not voisine(liste[i-1], liste[i]):
            return False
    return True
```

3. La preuve de terminaison de la boucle est apportée par l'incrémentation de l'indice `i` à chaque itération de la boucle, qui conduira à rendre faux la condition de continuation de la boucle (`i < len(cases) and possible`). En effet `i` dépassera obligatoirement `len(cases)` à un moment donné.

4.

```
def echappe(cases, laby):
    if not cases[0] == (0, 0):
        return False        # La première case est l'entrée.
    if not cases[-1] == (len(laby-1), len(laby-1)):
        return False        # La première case est l'entrée.
    if not adjacentes(cases):
        return False        # Les cases sont adjacentes.
    if not teste(cases, laby):
        return False        # Le chemin ne percute pas de mur.
    return True
```