Correction 1.

1. Preliminaires

(a) Aucune des matrices ne fonctionne ! `maze4` n'a pas les bons murs. `maze1` n'a pas d'entrée. `maze2` interverti entrée et sortie. `maze3` à la sortie mal placée.

(b) La bonne réponse est `chemin1`. `chemin1` est dans le mauvais sens. `chemin3` et `chemin4` intervertissent lignes et colonnes.

(c)
```
def entreeSortie(maze):
    n,p=np.shape(maze)
    for i in range(n):
        for j in range(p):
            if maze[i,j]==2:
                E=[i,j]
            if maze[i,j]==3:
                S=[i,j]
    return([E,S])
```

2. Recherche de chemin dans un labyrinthe parfait

(a)
```
def vide(lab, x, y):
    n, p = np.shape(lab)
    if 0<= x < p and 0 <= y < n and lab[x,y] in [0,2,3]:
        return True
    else:
        return False
```

(b)
```
def voisinsNonVisites(lab, i, j):
    liste = []
    cases = [[i+1,j], [i-1,j], [i,j-1], [i,j+1]]
    for k in range(3)
        x = cases[k][0]
        y = cases[k][1]
        if vide(lab,x,y):
            liste.append([x,y])
    return liste
```

(c) On peut utiliser l'instruction `chemin.pop()` qui supprime le dernier élément d'une liste.

(d)
```
def cheminSolution(lab):
    entree , sortie = entreeSortie(lab)
    i, j = entree
    chemin = [entree]
    lab[i,j] = 4
    while [i, j] != sortie :
        liste = voisinsNonVisites(lab, i, j)
        if len(liste) > 0:
            i, j = rd.choice(liste)
            lab[i,j] =4
            chemin.append([i,j])
        else:
            chemin.pop()
```

```
14              i, j = chemin[-1]
15      return chemin
```

3. Compression de chemins

(a) Le chemin-solution compressé dans le labyrinthe 1 est le suivant

```
[[7,0],[0,0],[0,2],[4,2],[4,10],[0,10] ]
```

(b)
```
2       return [chemin[k+1][0]-chemin[k][0], chemin[k+1][1]-chemin[k][1]]
```

(c)
```
2       dir = direction(chemin, 0)
3       chemin_compresse = [chemin[0]]
4       for k in range(1,len(chemin)):
5           if dir != direction(chemin, k):
6               chemin_compresse.append(chemin[k])
7               dir = direction(chemin, k)
8       chemin_compresse.append(chemin[k])
9       return chemin_compresse
```