```python
# Code Python du sujet 2
# Laure Gonnord, février/mars 2019
# + éléments de correction
# d'après une correction de Jean-Pierre Becirspahic (jp.becir@info-llg.fr).

from random import *

# Fonctions données
# il faut bien créer une valeur spéciale pour mettre dans le tableau;

def creerTableau(n):
    return [None] * n


def entierAleatoire(k):
    return (randrange(1, k+1))

# section 1 : ensembles sans redondance


def creerEnsVide(n):
    ens = creerTableau(n+1)
    ens[0] = 0
    return ens


def estDansEns(tab, x):
    for i in range(1, tab[0]+1):
        if tab[i] == x:
            return True
    return False

#Q3
def ajouteDansEns(tab, x):
    if not estDansEns(tab, x):
        tab[0] += 1
        tab[tab[0]] = x

#Q4 lorsque la structure est pleine et que l'élément n'y figure pas, la derniere ligne de
#la fonction précédente provoque une erreur._
#En considérant que l'affectation d'une valeur à une case d'un tableau est de coût constant, le coût
#de cette fonction est celle de estDansTab, ie O(n)

# Section 2

#Q4
plan1 = [[5, 7],
         [2, 2, 3, None, None],
         [3, 1, 3, 5, None],
         [4, 1, 2, 4, 5],
         [2, 3, 5, None, None],
         [3, 2, 3, 4, None]]


plan2 = [[5, 4],
         [1, 2, None, None, None],
         [3, 1, 3, 4, None],
         [1, 2, None, None, None],
         [2, 2, 5, None, None],
         [1, 4, None, None, None]]

print(plan1)
print(plan2)

# Q5 attention à bien utiliser les fonctions précédentes.
def creerPlanSansRoute(n):
    tab = creerEnsVide(n)
    tab[0] = [n, 0]
    for i in range(1, n+1):
        tab[i] = creerEnsVide(n-1)
    return tab

print(creerPlanSansRoute(3))

#Q6 attention au deuxième argument du range.
def estVoisine(plan, x, y):
    for i in range(1, plan[x][0] + 1):
        if plan[x][i] == y:
```

```
            return True
    return False


print(estVoisine(plan1,3,5))
print(estVoisine(plan2,3,1))
#true, false, cf les dessins de l'énoncé.


#Q7 ne pas faire == False, mais utiliser not
def ajouteRoute(plan, x, y):
    if not estVoisine(plan, x, y):
        plan[0][1] += 1
        ajouteDansEns(plan[x], y)
        ajouteDansEns(plan[y], x)

ajouteRoute(plan1,1,4)
print(estVoisine(plan1,4,1)) #true now

# Q8 si vous utiliser plusieurs fois la même constante, et qu'en plus elle est nommée
# dans l'énoncé, définissez-la!
def afficheToutesLesRoutes(plan):
    n, m = plan[0]
    print('Ce plan contient ', m, ' route(s) :', end='')
    for x in range(1, n+1):
        for i in range(1, plan[x][0]+1):
            y = plan[x][i]
            if x < y:
                print(' (', x, '-', y, ')', end='')
    print('\n')
# n tableaux à parcourir, la somme des longueurs de celles-ci = 2m, donc O(n+m)

afficheToutesLesRoutes(plan2)


# Partie Recherche de chemins arc-en-ciel

# Q9 regardez l'ordre de mes affectations, cela évite des tests inutiles!
def coloriageAleatoire(plan, couleur, k, s, t):
    for i in range(1, plan[0][0]+1):
        couleur[i] = entierAleatoire(k)
    couleur[s] = 0
    couleur[t] = k+1

# Q10 rien à dire
def voisinesDeCouleur(plan, couleur, i, c):
    lst = creerEnsVide(plan[0][0])
    for j in range(1, plan[i][0]+1):
        if couleur[plan[i][j]] == c:
            ajouteDansEns(lst, plan[i][j])
    return lst

# Q11 j'aurais pu utiliser la fonction précédente, faites ce que je dis, etc.
def voisinesDeLEnsDeCouleur(plan, couleur, ens, c):
    lst = creerEnsVide(plan[0][0])
    for i in range(1, ens[0]+1):
        for j in range(1, plan[ens[i]][0]+1):
            if couleur[plan[ens[i]][j]] == c:
                ajouteDansEns(lst, plan[ens[i]][j])
    return lst

#complexité O(n(n+m) avec une analyse similaire à la précédente.

#Q12 là encore, attention aux bornes!
def existeCheminArcEnCiel(plan, couleur, k, s, t):
    lst = creerEnsVide(plan[0][0])
    ajouteDansEns(lst, s)
    for c in range(1, k+2):
        lst = voisinesDeLEnsDeCouleur(plan, couleur, lst, c)
        if lst[0] == 0:
            return False
    return True

# à partir de là personne n'a composé, je vais vite.
def existeCheminSimple(plan, k, s, t):
    couleur = creerEnsVide(plan[0][0])
    for _ in range(k**k):
        coloriageAleatoire(plan, couleur, k, s, t)
```

```
        if existeCheminArcEnCiel(plan, couleur, k, s, t):
            return True
    return False

def voisinesDeLEnsDeCouleur2(plan, couleur, liste, c):
    lst = creerEnsVide(plan[0][0])
    for i in range(1, liste[0]+1):
        for j in range(1, plan[liste[i][-1]][0]+1):
            if couleur[plan[liste[i][-1]][j]] == c:
                ajouteDansEns(lst, liste[i] + [plan[liste[i][-1]][j]])
    return lst

def existeCheminArcEnCiel2(plan, couleur, k, s, t):
    ens = creerEnsVide(plan[0][0])
    ajouteDansEns(ens, [s])
    for c in range(1, k+2):
        ens = voisinesDeLEnsDeCouleur2(plan, couleur, ens, c)
        if ens[0] == 0:
            return False, []
    return True, ens[1]


def existeCheminSimple2(plan, k, s, t):
    couleur = creerEnsVide(plan[0][0])
    for _ in range(k**k):
        coloriageAleatoire(plan, couleur, k, s, t)
        r, lst = existeCheminArcEnCiel2(plan, couleur, k, s, t)
        if r:
            return lst
        return False

print(existeCheminSimple(plan2, 2, 1, 5))
print(existeCheminSimple2(plan2, 2, 1, 5))
```