corrigé

Le jeu de Snort

On commence par calculer la liste des 1000 premières valeurs de la suite (u_n) :

```
u = [42]
for _ in range(1, 1000):
    u.append((19999999 * u[-1]) % 19999981)
```

Question 1.

```
def genereGraphe(n, k):
    G = [[] for s in range(n)]
    for p in range(n):
        for q in range(p + 1, n):
            if (u[p + q]) % (10000 + n) < k:
                G[p].append(q)
                G[q].append(p)
    return G
```

La fonction qui suit compte le nombre d'arêtes d'un graphe non orienté :

```
def nbAretes(G):
    somme = 0
    for s in range(len(G)):
        somme += len(G[s])
    return somme // 2
```

Question 2.

```
def estValide(G, couleur):
    for p in range(len(G)):
        for q in G[p]:
            if (couleur[p], couleur[q]) in ((1, 2), (2, 1)):
                return False
    return True
```

Question 3.

```
def coupsPossiblesRouge(G, couleur):
    lst = []
    for p in range(len(G)):
        if couleur[p] == 0:
            if 2 not in [couleur[q] for q in G[p]]:
                lst.append(p)
    return lst

def coupsPossiblesBleu(G, couleur):
    lst = []
    for p in range(len(G)):
        if couleur[p] == 0:
            if 1 not in [couleur[q] for q in G[p]]:
                lst.append(p)
    return lst
```

Question 4.

```python
def encode(couleur):
    m = 0
    for k in range(len(couleur)):
        m += couleur[k] * 3 ** k
    return m

def decode(m):
    if m == 0:
        return [0]
    couleur = []
    while m > 0:
        couleur.append(m % 3)
        m //= 3
    return couleur
```

Question 5.

```python
def partie(G):
    n = len(G)
    couleur = [0 for _ in range(n)]
    joueur = 1
    coups = []
    while True:
        if joueur == 1:
            lst = coupsPossiblesRouge(G, couleur)
            if lst == []:
                return('Bleu', coups)
            c = min(lst)
            couleur[c] = 1
            coups.append(c)
            joueur = 2
        else:
            lst = coupsPossiblesBleu(G, couleur)
            if lst == []:
                return('Rouge', coups)
            c = min(lst)
            couleur[c] = 2
            coups.append(c)
            joueur = 1
```

Question 6.

```
def rougeGagne(G, couleur):
    code = encode(couleur)
    if code not in positionsGagnantes:
        lst = coupsPossiblesRouge(G, couleur)
        b = False
        for s in lst:
            color = couleur.copy()
            color[s] = 1
            if not bleuGagne(G, color):
                b = True
        positionsGagnantes[code] = b
    return positionsGagnantes[code]

def bleuGagne(G, couleur):
    code = encode(couleur)
    if code not in positionsGagnantes:
        lst = coupsPossiblesBleu(G, couleur)
        b = False
        for s in lst:
            color = couleur.copy()
            color[s] = 2
            if not rougeGagne(G, color):
                b = True
        positionsGagnantes[code] = not b
    return not positionsGagnantes[code]
```