Corrigé du DS d'informatique n°3 du 06/05/2025

1.
```
def creerGrille(largeur, hauteur):
    return [ [X for _ in range(hauteur)] for _ in range(largeur)  ]
```

   Autre possibilité

```
def creerGrille(largeur, hauteur):
    grille = [ ]
    for _ in range(largeur):
        ligne = [X] * hauteur
        grille.append(ligne)
    return grille
```

2.
```
def tailleGrille(grille):
    return len(grille), len(grille[0])
```

3.
```
def afficheGrille(grille):
    largeur = len(grille)
    y = len(grille[0]) - 1
    while y >= 0:
        for x in range(largeur):
            afficheCouleur( grille[x][y] )
        y -= 1
        nouvelleLigne()
```

4. `bloc = grille[x][y:y+k]`

5.
```
def grilleLibre(grille, k):
    largeur, hauteur = tailleGrille(grille)
    for x in range(largeur):
        bloc = [grille[x][hauteur-1-i] for i in range(k)]
        if bloc == [X]*k:
            return True
    return False
```

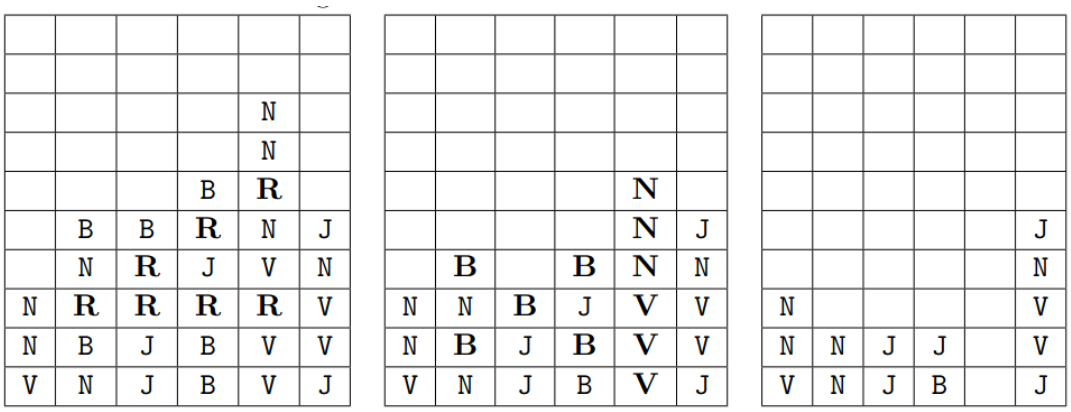
6. Elle effectue exactement `k*largeur` tests d'égalité et le reste est dominé par cette quantité. Elle effectue donc un nombre d'opérations élémentaires en $O($ `k*largeur` $)$.

7.
```
def descente(grille, x, y, k):
    segment = [grille[x][y+i] for i in range(k)]
    if y >= 1 and grille[x][y-1] == X:
        grille[x][y-1] = segment[0]
        for i in range(k-1):
            grille[x][y+i] = segment[i+1]
        grille[x][y+k-1] = X
```

8.
```python
def deplacerBarreau(grille, x, y, k, direction):
    largeur, hauteur = tailleGrille(grille)
    if x + direction < largeur and x + direction >= 0:
        if grille[x+direction][y:y+k] == [X]*k:
            grille[x+direction][y:y+k], grille[x][y:y+k] = grille[x][y:y+k], grille[x+direc
```

9.
```python
def permuterBarreau(grille, x, y, k):
    segment = grille[x][y:y+k]
    grille[x][y:y+k] = segment[k-1:k] + segment[0:k-1]
```

10. On obtient les trois configurations successives suivantes Soit un score de $(2+2)+(1+1+1+1)=8$ points.



11.
```python
def detecteAlignement(rangee):
    n = len(rangee)
    marking = [False] * n
    score = 0

    couleur_actuelle = rangee[0]
    index_debut = 0
    compteur = 1

    for i in range(1, n):
        couleur = rangee[i]

        if couleur == couleur_actuelle:
            compteur += 1
        else:
            # Changement de couleur détecté : on traite la séquence qui vient de finir
            if couleur_actuelle != X and compteur >= 3:
                score += (compteur - 2)
                # On marque les indices de la séquence précédente
                for j in range(index_debut, i):
                    marking[j] = True

            # Réinitialisation pour la nouvelle séquence
            couleur_actuelle = couleur
            index_debut = i
            compteur = 1
```

```
    # Traitement de la toute dernière séquence après la fin de la boucle
    if couleur_actuelle != X and compteur >= 3:
        score += (compteur - 2)
        for j in range(index_debut, n):
            marking[j] = True

    return (marking, score)
```

Et la version Charles Bourcier, plus concise et élégante, avec pour seul soucis qu'elle accède plusieurs fois à chaque case.

```
def detecteAlignement(rangee):
    n = len(rangee)
    marking = [False]*n
    score = 0
    for i in range(1,n-1):
        if rangee[i] != X:
            if rangee[i-1] == rangee[i] == rangee[i+1]:
                for j in range(-1,2):
                    marking[i+j] = True
                score += 1
    return marking, score
```

12. Les listes sont mutables donc ceci ne fait pas une copie. De plus comme c'est une liste de liste de couleurs il faut copier chaque sous liste ou utiliser deepcopy.

```
largeur, hauteur = tailleGrille(grille)
g = [[grille[i][j] for j in range(hauteur)] for i in range(largeur)]
```

Ou encore

```
from copy import deepcopy
g = deepcopy(grille)
```

13.
```
def scoreRangee(grille, g, i, j, dx, dy):
    largeur, hauteur = tailleGrille(grille)
    x, y = i, j
    rangee = [ ]
    while 0 <= x and x < largeur and 0 <= y and y < hauteur:
        rangee.append(grille[x][y])
        x += dx
        y += dy
    marking, score = detecteAlignement(rangee)
    n = len(rangee)
    for p in range(n):
        if marking[p]:
            g[i+p*dx][j+p*dy] = X

    return score
```

14.
```
def effaceAlignement(grille):
    largeur, hauteur = tailleGrille(grille)
```

```
    score = 0

    # Vraie copie de la grille
    g = [[grille[i][j] for j in range(hauteur)] for i in range(largeur)]

    # Test des lignes
    dx, dy = 1, 0
    for j in range(hauteur):
        score += scoreRangee(grille, g, 0, j, dx, dy)

    # Test des colonnes
    dx, dy = 0, 1
    for i in range(largeur):
        score += scoreRangee(grille, g, i, 0, dx, dy)

    # Test des diagonales montant vers la droite
    dx, dy = 1, 1
    for i in range(largeur):
        score += scoreRangee(grille, g, i, 0, dx, dy)
    for j in range(1, hauteur):
        score += scoreRangee(grille, g, 0, j, dx, dy)

    # Test des diagonales descendant vers la droite
    dx, dy = 1, -1
    for i in range(largeur):
        score += scoreRangee(grille, g, i, hauteur - 1, dx, dy)
    for j in range(hauteur - 1):
        score += scoreRangee(grille, g, 0, j, dx, dy)

    return g, score
```

15.
```
def tassementGrille(grille):
    largeur, hauteur = tailleGrille(grille)
    for x in range(largeur):
        k = 0
        for y in range(hauteur):
            if grille[x][y] == X:
                k += 1
            elif k >= 1:
                grille[x][y-k] = grille[x][y]
                grille[x][y] = X
```

16.
```
def calculScore(grille):
    g = [[grille[i][j] for j in range(hauteur)] for i in range(largeur)]
    largeur, hauteur = tailleGrille(g)
    scoreTotal = 0
    score = 1
    while score != 0:
        g, score = effaceAlignement(g)
        tassementGrille(g)
        scoreTotal += score
    return (g, scoreTotal)
```