Algorithmie
Epreuve pratique
Algorithme des K plus proches voisins

1 Contexte

1.1 Algorithme Naïf

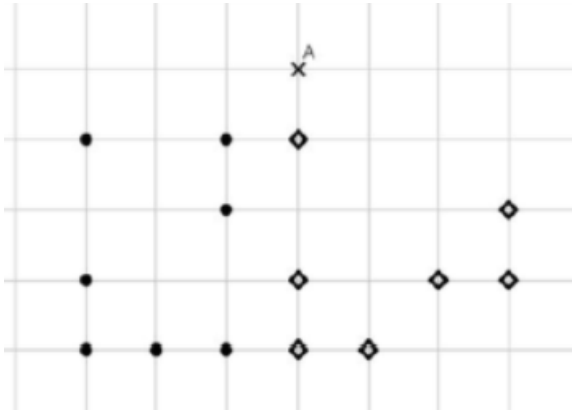
Ecrire en langage Python une fonction nommée **knn(base,k,inconnu)** qui prend en argument une base d'apprentissage, un entier K, et un élément appelé inconnu. L'algo renvoie l'étiquette majoritaire.

Rappel du principe de l'algorithme knn :

```
1 FONCTION KNN(base : base d'apprentissage, K :entier, inconnu : nouvel élément)
2 |   POUR Pour chaque donnée dans la base d'apprentissage FAIRE
3 |   |   Calculez la distance entre l'inconnu et la donnée courante.
4 |   |   Mémoriser cette distance et la donnée associée dans une liste de couple
  |   |    (distance, donnée).
5 |   Triez la liste distance_ Donnée du plus petit au plus grand sur les distances.
6 |   Choisissez les k premières entrées de cette liste.
7 |   Obtenir les étiquettes des k données de la base d'apprentissage sélectionnées.
8 |   Renvoyer l'étiquette majoritaire
```

1.2 Présentation d'un problème, tiré d'un sujet zéro

Dans le quadrillage ci-dessous 14 points sont dessinés, dont 7 de la classe C1, avec des ronds noirs, et 7 de la classe C2, avec des losanges. On introduit un nouveau point A, dont on cherche la classe à l'aide d'un algorithme des k plus proches voisins pour la distance géométrique habituelle, en faisant varier la valeur de k parmi 1, 3 et 5. Quelle est la bonne réponse (sous la forme d'un triplet de classes pour le triplet (1,3,5) des valeurs de k) ?



1.3 Spécifications

Écrire une fonction Python **KNN(base,k,nouveau)** avec :

- base est la base d'apprentissage
- k un entier (Le nombre de plus proches voisins)
- nouveau est le nouvel élément à étudier

La fonction renverra l'étiquette majoritaire (ronds noirs ou losanges).

☞ On utilisera ici la distance classique (distance euclidienne) entre deux points du plan.
☞ On utiliser le dictionnaire suivant comme base d'apprentissage :

```
base = {(1,0):'rond',(2,0):'rond',(3,0):'rond',(1,1):'rond',
(1,3):'rond',(3,2):'rond',(3,3):'rond',(4,0):'losange',(5,0):'losange',
(4,1):'losange',(6,1):'losange',(7,1):'losange',(7,2):'losange',
(4,3):'losange'}
```

☞ nouveau sera donc la donnée (4,4).

2 Résolution

2.1 Un programme Python

```
def distance_cible(donnees,cible):
    table = []
    for i,j in donnees.items():
        d = sqrt((i[0]-cible[0]) ** 2 + (i[1] - cible[1]) ** 2)
        table.append([i,j,d])
    return table

def majoritaire(L):
    rond,losange = 0,0
    for i in L:
        if i == 'rond':
            rond += 1
        else:
            losange += 1
```

```
    if rond > losange:
        return 'rond'
    else:
        return 'losange'

def knn(base,k,nouveau):
    base_avec_distance = distance_cible(base,nouveau)
    base_avec_distance_triee = sorted(base_avec_distance, key = lambda x : x[2])
    proches_voisins = []
    for i in range(k):
        proches_voisins.append(base_avec_distance_triee[i][1])
    return majoritaire(proches_voisins)
```