```python
#  Algorithme kNN de la page 4 du cours chapitre 10.
#  Auteur : ...

import math

def distance_euclidienne(xA, yA, xB, yB):
    d = ...
    return d
```

```
def calcule_distances(x, y, liste):
    """
    Prend en paramètre la liste du jeu de données.
    Par exemple
    liste = [[15, 50, 'rouge'], [20, 10, 'rouge'], [25, 15, 'rouge'], \
         [30, 18, 'bleu'], [40, 40, 'rouge'], [42, 4, 'bleu'], \
         [55, 20, 'bleu'], [60, 5, 'rouge'], [65, 13, 'rouge'], \
         [70, 39, 'rouge']]

    Renvoie la liste de doublets
    [(distance entre P et le 1er elt, classe du 1er elt),\
     (distance entre P et le 2e elt, classe du 2e elt), ..., ... ].
    Par exemple [(320.2, 'bleu'), (432.85, 'rouge'), ... ]

    Paramètres :
    ------------
        x, y : de type flottant
               Ce sont les coordonnées du point inconnu P à classer.

       liste : de type liste
               C'est le jeu de données.

    Renvoie :
    ---------
        liste_doublets : de type liste
                         C'est la liste des doublets



    """

    liste_doublets = []  #  Initialisation par une liste vide.

    ...
```

```
def tri(tableau):
    """
    prend en paramètre la liste tableau des doublets
    Par exemple [(320.2, 'bleu'), (432.85, 'rouge'), ... ]

    Renvoie le tableau trié selon l'ordre des distance croissantes.
     Par exemple [(8.25, 'bleu'), (9.35, 'rouge'), ... ]

    Paramètres :
    ------------
       tableau : de type liste
               C'est la liste des doublets non triée renvoyée par la
               fonction calcule_distances(x, y, liste).


    Renvoie :
    ---------
        liste_doublets_triee : de type liste
                C'est la liste des doublets triée

    """

    #  On trie la liste_distances dans l'ordre des distances croissantes.

    ...
```

```
def classe(k, tableauTrie):
    """
    Prend en paramètre :
       - Le nombre k des voisins à prendre en compte.
       - Le tableau des doublets triés.
         Par exemple [(8.25, 'bleu'), (9.35, 'rouge'), ... ]

    Renvoie le dictionnaire dico_effectifs des classes sous la forme
    {'classe 1': effectif 1, 'classe 2': effectif 2}
    Exemple pour k = 3 voisins à prendre en compte : {'rouge': 1, 'bleu': 2}

    Paramètres :
    ------------
        k : de type entier.
        tableauTrie : de type liste
               C'est la liste des doublets triée renvoyée par la
               fonction tri(tableau).

    Renvoie :
    ---------
        dico_effectifs : de type dictionnaire
                         C'est le dictionnaire des effectifs parmi les
                         k premiers doublets de liste_distances_triee
    """

    dico_effectifs = {}    # Initialisation par un dictionnaire vide.




#  Programme
#  On saisit la liste du jeu de données
liste = [[15, 50, 'rouge'], [20, 10, 'rouge'], [25, 15, 'rouge'], \
         [30, 18, 'bleu'], [40, 40, 'rouge'], [42, 4, 'bleu'], \
         [55, 20, 'bleu'], [60, 5, 'rouge'], [65, 13, 'rouge'], \
         [70, 39, 'rouge']]

#  On calcule la liste des doublets
# (distance entre P(52, 7) et point du jeu de données, classe du point)
liste_distances = calcule_distances(52, 7, liste)

#  On trie la liste des doublets par ordre croissant des distances.
liste_distances_triee = tri(liste_distances)

#  On calcule le dictionnaire qui donne l'effectif pour chaque classe en se
#  limiatnt aux k = 3 plus proches voisins de P.
mon_dico = classe(3, liste_distances_triee)

print(mon_dico)
```