**FONCTIONS**

distance(perso1, perso2):

"""

Calcule la distance euclidienne entre deux personnages

Entrée: Les deux personnages (dictionnaires)
Sortie: La distance euclidienne entre les deux personnages (flottant)
"""

ajouter_distances(table, personnage_inconnu):

"""

Ajoute la distance par rapport à un personnage cible sur tous les personnage

Entrée: Un tableau de dictionnaires de chaque personnages
Le personnage de référence pour le calcul des distances (dictionnaire)
Sortie: Le tableau de dictionnaires avec les distances ajoutées
"""

**VARIABLES**

perso1, perso2, personnage_inconnu, maisons, voisin : dictionnaires
table, voisins : listes
max, meilleur, n, nombre_tests, bien_deviné : entiers
meilleur_maison, maison: chaîne de caractères

**DÉBUT**

**DÉFINIR FONCTION** meilleur_maison(voisins):

"""

Renvoie la maison apparaissant le plus dans la liste des plus proches voisins

Entrée: Liste des plus proches voisins
Sortie: La maison apparaissant le plus parmi la liste des voisins
(chaîne de caractères)
"""

```
maisons ← {}
POUR voisin DANS voisins:
    SI voisin['Poste'] DANS maisons:
        maisons[voisin['House']] ← maisons[voisin['House']] + 1
    SINON:
        maisons[voisin['House']] ← 1
    FIN SINON
FIN POUR
max ← 0
POUR maison, n DANS items(maisons):
    SI n > max:
        max ← n
        meilleur_maison ← maison
    FIN SI
FIN POUR
RENVOYER meilleur_maison
```

**FIN FONCTION**

```
DÉFINIR FONCTION données_test(table):
    """
    Renvoie les 3 quarts des éléments de la liste en enlevant aléatoirement le
    quart

    Entrée: Une liste de personnages
    Sortie:  Une liste contenant les 3/4 de la liste originelle
    """

    personnages_test ← []
    copie_personnages ← table[:]
    POUR _ DE 0 À taille(copie_personnages // 4) - 1:
        ajouter un personnage aléatoire de copie_personnages à
        personnages_test en le supprimant de copie_personnages
    FIN POUR
    RENVOYER joueurs_test, copie_joueurs
FIN FONCTION

DÉFINIR FONCTION meilleur_k(table):
    """
    Calcule le nombre de voisins donnant le résultat le plus précis pour une
    personne inconnue

    Entrée: Une liste de personnages
    Sortie: Le nombre de voisins le plus précis (entier)
    """

    nombre_tests ← 100
    meilleur ← 0
    POUR k DE 1 À 19:
        bien_deviné ← 0
        POUR _ DE 0 À taille(nombre_tests) -1:
            personnages_test, personnage_de_référence ←
            données_test(table)
            POUR cible DANS personnages_test:
                personnage_de_référence ← ajouter_distances(
                personnage_de_référence, cible)
                voisins ← tri de personnage_de_référence en fonction
                de leur distance dans l'ordre croissant
                SI meilleur_maison(voisins[:k]) = cible['Poste']:
                    bien_deviné +← 1
                FIN SI
            FIN POUR
        SI bien_deviné > meilleur:
            meilleur_k ← k
            meilleur ← bien_deviné
        FIN SI
    FIN POUR
    RENVOYER meilleur_k
FIN FONCTION
```

**DÉFINIR FONCTION** maison(table, personnage, k):

"""

Calcule la meilleur maison pour un personnage inconnu en fonction
du nombre de voisins sélectionnés

Entrée: Une liste de personnages
Le personnage pour lequel on détermine sa maison (dictionnaire)
Le nombre de voisins pris en compte pour le calcul (entier)
Sortie: La meilleur maison pour le personnage (chaîne de caractères)
Ses k plus proches voisins (liste)

"""

personnages ← ajouter_distances(table, personnage)
voisins ← tri de voisins en fonction de la distance dans l'ordre croissant
**RENVOYER** meilleur_maison(voisins[:k]), voisins[:k]

**FIN FONCTION**

**FIN**