EXERCICE 1 (5 points)

Programmer une fonction `renverse`, prenant en paramètre une chaîne de caractères non vide, `mot`, et qui renvoie une chaîne de caractères en inversant ceux de la chaîne `mot`.

Exemple :

```
print(renverse("informatique") )  #"euqitamrofni"
```

EXERCICE 2 (5 points)

Le jeu du « plus ou moins » consiste à deviner un nombre entier choisi entre 1 et 99.
Un élève de NSI décide de le coder en langage Python de la manière suivante :

- le programme génère un nombre entier aléatoire compris entre 1 et 99 ;
- si la proposition de l'utilisateur est plus petite que le nombre cherché, l'utilisateur en est averti. Il peut alors en tester un autre ;
- si la proposition de l'utilisateur est plus grande que le nombre cherché, l'utilisateur en est averti. Il peut alors en tester un autre ;
- si l'utilisateur trouve le bon nombre en 10 essais ou moins, il gagne ;
- si l'utilisateur a fait plus de 10 essais sans trouver le bon nombre, il perd.

La fonction `randint` du module `random` est utilisée.
Si `a` et `b` sont des entiers tels que `a <= b`, `randint(a, b)` renvoie un nombre entier compris entre `a` et `b` inclus.

Compléter le code ci-dessous et le tester :

```
from random import randint

def plus_ou_moins():
    nb_mystere = randint(1, ...)
    nb_test = int(input("Proposez un nombre entre 1 et 99 : "))
    compteur = ...
    while nb_mystere != ... and compteur < ...:
        compteur = compteur + ...
        if nb_mystere ... nb_test:
            nb_test = int(input("Trop petit ! Testez encore :
        "))
        else:
            nb_test = int(input("Trop grand ! Testez encore :
        "))
    if nb_mystere == nb_test:
        print("Bravo ! Le nombre était ", ...)
        print("Nombre d'essais: ", ...)
    else:
        print("Perdu ! Le nombre était ", ...)

plus_ou_moins()
```

EXERCICE 3 (5 points)

Écrire une fonction `recherche(caractere, chaine)` qui prend en paramètres `caractere`, un unique caractère (c'est-à-dire une chaîne de caractère de longueur 1), et `chaine`, une chaîne de caractères. Cette fonction renvoie le nombre d'occurrences de `caractere` dans `chaine`, c'est-à-dire le nombre de fois où `caractere` apparaît dans `chaine`.
Exemples :

```
print(recherche('e', "sciences")) #2
print(recherche('i', "mississippi")) #4
print(recherche('a', "mississippi"))#0
```

EXERCICE 4 (5 points)

On considère des mots à trous : ce sont des chaînes de caractères contenant uniquement des majuscules et des caractères `'*'`. Par exemple `'INFO*MA*IQUE'`, `'***I***E**'` et `'*S*'` sont des mots à trous.
Programmer une fonction `correspond` qui :

- prend en paramètres deux chaînes de caractères `mot` et `mot_a_trous` où `mot_a_trous` est un mot à trous comme indiqué ci-dessus,
- renvoie :
  - `True` si on peut obtenir `mot` en remplaçant convenablement les caractères `'*'` de `mot_a_trous`.
  - `False sinon.`

Exemples :

```
print(correspond('INFORMATIQUE', 'INFO*MA*IQUE')) #True
print(correspond('AUTOMATIQUE', 'INFO*MA*IQUE')) #False
print(correspond('STOP', 'S*')) #False
print(correspond('AUTO', '*UT*')) #True
```