Exercices d'algorithmique n°1

Exercice 1 : Écrire une fonction `double` qui prend un nombre `x` et qui renvoie le double du nombre `x`

```
def double(x):
```

Exercice 2 : Écrire une fonction `carre` qui prend un nombre `x` et qui renvoie le carré du nombre `x`

```
def carre(x):
```

Exercice 3 : Écrire une fonction `prod_pos` qui prend des nombres `x` et `y`, et qui renvoie un booléen qui indique si le produit de `x` par `y` est supérieur ou égal à 0.

```
def prod_pos(x, y):
```

Exercice 4 : Écrire une fonction `mention` qui prend un nombre `note` et qui affiche `Echec`, `Pas de mention` ou `Mention`, si note < 10, $10 \leq$ note < 12 ou note ≥ 12.

```
def mention(note):
```

Exercice 5 : Écrire une fonction `somme_carres` qui prend un entier `n` et qui renvoie la somme des carrés de 1 à `n` inclus. Par exemple : `somme_carres(3)` $= 1^2 + 2^2 + 3^2 = 14$.

```
def somme_carres(n):
```

Exercice 6 : Écrire une fonction `longueur` qui prend une chaîne de caractères `texte` et qui renvoie un entier correspondant au nombre de symboles de `texte`. On n'utilisera pas la fonction `len`.

```
def longueur(texte):
```

Exercice 7 : Écrire une fonction `appartient` qui prend une chaîne de caractères `texte` et un symbole `symbole`, et qui renvoie un booléen indiquant si `symbole` est contenu dans `texte`.

```
def appartient(symbole, texte):
```

Exercice 8 : Écrire une fonction `compter` qui prend une chaîne de caractères `texte` et un symbole `symbole`, et qui renvoie un entier correspondant au nombre d'occurrences de `symbole` dans `texte`. On n'utilisera pas la fonction `count`.

```
def compter(symbole, texte):
```

Exercice 9 : Écrire une fonction `indice` qui prend une chaîne de caractères `texte` et un symbole `symbole`, et qui renvoie un entier correspondant à l'indice de la première occurrence de `symbole` dans `texte`. S'il n'y en a pas, la fonction renvoie -1. On n'utilisera pas `index`.

```
def indice(symbole, texte):
```

Exercice 10 : Écrire une fonction `nombres_de_multiples` qui prend 3 entiers `k1`, `k2` et `n`, et qui renvoie le nombre d'entiers qui sont multiples de `k1` ou de `k2` et qui sont strictement inférieurs à `n`. On rappelle que `m` est un multiple de `d` si `m%d` vaut `0`.

```
def nombres_de_multiples(k1, k2, n):
```