EZ Language
Fonctions sur les chaînes de caractères

Rappel sur les opérateurs
(voir fichier sur les opérateurs)

Les opérateurs suivants seront définis pour les chaînes de caractères :

- L'opérateur d'affectation `=` qui affectera une nouvelle valeur à la chaîne de caractères, remplaçant son contenu actuel.
- L'opérateur d'accès `[]` qui retournera une référence sur le caractère à la position spécifiée. On pourra parcourir une chaîne de la même manière qu'un tableau.
- L'opérateur de concaténation `+` qui permettra de concaténer la chaîne en partie gauche et la chaîne en partie droite.
- L'opérateur `+=` qui ajoutera les caractères à droite de l'opérateur à la fin de la chaîne à gauche de l'opérateur.
- L'opérateur de comparaison `==` qui retournera `true` si la chaîne à gauche de l'opérateur est égale à la chaîne à droite de l'opérateur, `false` sinon.
- L'opérateur de comparaison `!=` qui retournera `true` si la chaîne à gauche de l'opérateur est différente de la chaîne à droite de l'opérateur, `false` sinon.

Fonctions
Les paramètres entre [] sont optionnels.

Pour tous les exemples des méthodes ci-dessous, on utilisera la variable : `s is string`

length
- *Signature*

  length() return integer

- *Syntaxe*

```
s.length()
```

Retourne la longueur de la chaîne s.

toUpperCase
- *Signature*

  toUppercase() return string

- *Syntaxe*

```
s.toUpperCase()
```

Retourne une copie de la chaîne s avec tous les caractères en majuscule.

toLowerCase

- *Signature*

toLowerCase() return string

- *Syntaxe*

```
s.toLowerCase()
```

Retourne une copie de la chaîne avec tous les caractères en minuscule.

substring

- *Signature*

substring(start is int[, length is int]) return string

- *Syntaxe*

```
start, length are integer
s.substring(start)
s.substring(start, length)
```

Retourne une nouvelle chaîne qui est la sous-chaîne de s à partir du caractère start et de longueur length. Si length n'est pas précisé, par défaut length sera égal à s.length().

split

- *Signature*

split(pattern is string) return vector of string
split(pattern is regex) return vector of string

- *Syntaxe*

```
s.split(";")
```

Découpe la chaîne s selon la chaîne ou l'expression régulière passée en paramètre et retourne un vecteur de string contenant les sous-chaînes trouvées.

join

- *Signature*

join(iterable is array of string) return string
join(iterable is vector of string) return string
join(iterable is list of string) return string
join(iterable is set of string) return string

- *Syntaxe*

```
my_tab is array[1..5] of string
s.join(my_tab)
```

Rassemble les éléments d'un tableau, d'un vecteur, d'une liste ou d'un set en une chaîne. Les éléments sont séparés par la chaîne de caractères s. Les éléments du conteneur passé en paramètre doivent obligatoirement être des chaînes de caractères.

strip

- *Signature*

strip([character_mask is string]) return string
strip([character_mask is regex]) return string

- *Syntaxe*

```
s.strip()
s.strip("abcd")
```

Retourne une copie de la chaîne s, à laquelle on a enlevé tous les caractères invisibles en début et en fin de chaîne. Si une chaîne de caractères est précisée dans les paramètres, alors tous les caractères **faisant partie** de cette chaîne seront également enlevés. Si une expression régulière est précisée, alors les séquences de s correspondant à cette expression seront supprimées.

NB : Les fonctions C++ concernant les regex et la classe match_results pourront être utilisées pour traduire cette fonction en C++.

replace

- *Signature*

replace(search is string, replacement is string) return string
replace(search is regex, replacement is string) return string

- *Syntaxe*

```
s.replace("ab", "#")
```

Remplace toutes les occurrences de `search` (ou correspondant à l'expression régulière passée en paramètre) par la chaîne `replacement` dans la chaîne s.

NB : Correspond à la fonction regex_replace en C++

contains

- *Signature*

contains(search is string) return boolean
contains(search is regex) return boolean

- *Syntaxe*

```
s.contains("toto")
```

Retourne vrai si la sous-chaîne `search` (ou si une sous-chaîne correspondant à l'expression régulière `search`) est présente dans la chaîne s.

NB : Correspond à la fonction regex_search en C++

find

- *Signature*
  find(search is string) return int

- *Syntaxe*
  `s.find("toto")`

Retourne la position de la première occurrence de la chaîne `search` (ou d'une séquence correspondant à l'expression régulière passée en paramètre) dans la chaîne s. Si un seul caractère correspond cela ne suffit pas, il faut que la chaîne entière corresponde (contrairement aux fonctions findFirstOf et findLastOf). Si la chaîne `search` n'est pas trouvée, la fonction renvoie –1.

findFirstOf

- *Signature*
  findFirstOf(pattern is string) return integer
  findFirstOf(pattern is regex) return integer

- *Syntaxe*
  `s.findFirstOf("aeiou")`

Retourne l'indice de la première occurrence d'un caractère **faisant partie** de pattern dans la chaîne s. Si aucun des caractères de pattern n'est trouvé, renvoie -1. Si une regex est passée en paramètre, retourne l'indice de la première occurrence de la séquence correspondante trouvée.

NB : Les fonctions C++ concernant les regex et la classe match_results pourront être utilisées pour traduire cette fonction en C++.

findLastOf

- *Signature*
  findLastOf(pattern is string) return integer
  findLastOf(pattern is regex) return integer

- *Syntaxe*
  `s.findLastOf("aeiou")`

Retourne l'indice de la dernière occurrence d'un caractère **faisant partie** de pattern dans la chaîne s. Si aucun des caractères de pattern n'est trouvé, renvoie -1. Si une regex est passée en paramètre, retourne l'indice de la première occurrence de la séquence correspondante trouvée.

NB : Les fonctions C++ concernant les regex et la classe match_results pourront être utilisées pour traduire cette fonction en C++.