Strings Functions

A Reminder on Operators :

(See file on operators)

The following operators will be defined for strings :

- The assignment operator = which assigns a new value to the string, replacing its current content.
- The access operator [] which returns a reference on the character at the specified position. You can iterate over a string in the same way as an array.
- The concatenation operator + which will concatenate a string in the left part and the string in the right part.
- The += operator adds the string to the right of the operator at the end of the string to the left of the operator.
- The comparison operator == that returns true if the string to the left of the operator is equal to the string to the right of the operator, otherwise it returns false.
- The comparison operator != Which returns true if the string to the left of the operator is different from the string to the right of the operator, it returns false otherwise.

Functions :

Parameters between [] are optional.

For all the examples of the methods below, the variable will be used : `s is string`

length :

- *Signature :*

  ```
  length() return integer
  ```

- *Syntax :*

  ```
  s.length()
  ```

Returns the length of the string s.

toUpperCase :

- *Signature :*

```
toUppercase() return string
```

- *Syntax :*

```
s.toUpperCase()
```

Returns a copy of the string s with all characters to uppercase.

toLowerCase :

- *Signature :*

```
toLowerCase() return string
```

- *Syntax :*

```
s.toLowerCase()
```

Returns a copy of the string s with all characters to lowercase.

substring :

- *Signature :*

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

- *Syntax :*

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

Returns a new string that is the substring of s from the start character and of length : length. If length is not specified, default length will be equal to s.length().

split :

- *Signature :*

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

- *Syntax :*

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

Cuts the string s according to the string or regular expression passed as a parameter and returns a string vector containing the substrings found.

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
```

- *Syntax :*

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

Joins the elements of an array, a vector, a list or a set into a string. The elements are separated by the string s. The elements of the container passed as a parameter must be strings of characters.

strip :

- *Signature :*

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

- *Syntax :*

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

Returns a copy of the string s, to which removed all the invisible characters at the beginning and end of string. If a character string is specified in the parameters, then all characters **being part of** that string will also be removed. If a regular expression is specified, then the sequences corresponding to this expression will be deleted.

NB : C++ functions for regex and match_results can be used to translate this function to C++.

replace :

- *Signature :*

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

- *Syntax :*

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

Replaces all occurrences of `search` (or matching the regular expression passed as a parameter) with the `replacement` string in string s.

NB : Corresponds to the regex_replace function in C ++.

contains :

- *Signature :*

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

- *Syntax :*

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

Returns true if the substring `search` (or if a substring corresponding to the regular expression `search`) is present in string s..

NB : Corresponds to the regex_search function in C++.

find :

- *Signature :*

```
find(search is string) return int
```

- *Syntax :*

```
s.find("toto")
```

Returns the position of the first occurrence of the `search` string (or a sequence corresponding to the regular expression passed as a parameter) in the string s. If only one character matches, that's is not enough, the entire string must match (unlike the findFirstOf and findLastOf functions). If the `search` string is not found, the function returns -1.

findFirstOf :

- *Signature :*

```
findFirstOf(pattern is string) return integer
findFirstOf(pattern is regex) return integer
```

- *Syntax :*

```
s.findFirstOf("aeiou")
```

Returns the index of the first occurrence of a character that **is part of** a pattern in the s string. If none of the pattern characters are found, returns -1. If a regex is passed as a parameter, returns the index of the first occurrence of the corresponding sequence found.

NB : C++ functions for regex and match_results can be used to translate this function into C++.

findLastOf :

- *Signature :*

```
findLastOf(pattern is string) return integer
findLastOf(pattern is regex) return integer
```

- *Syntax :*

```
s.findLastOf("aeiou")
```

Returns the index of the last occurrence of a character that **is part of** a pattern in the s string. If none of the pattern characters are found, returns -1. If a regex is passed as a parameter, returns the index of the last occurrence of the corresponding sequence found.

NB : C++ functions for regex and match_results can be used to translate this function into C++.