Manipulation de chaînes de caractères en OCaml

Les chaînes de caractères en OCaml sont représentées en mémoire comme des tableaux non mutables de caractères. Le type des chaînes de caractères est le type `string` qui est un alias pour `String.t`. Le module `String` fournit de nombreuses fonctions de manipulation des chaînes de carcatères.

Création. On peut fabriquer une chaîne de caractères :

- au moyen de sa description littérale : l'expression `"toto"` s'évalue en la chaîne de caractères `"toto"` de type `string` ;
- au moyen de la fonction `String.make : int -> char -> string` prenant en arguments un entier $n \geqslant 0$ et un caractère c et retournant la chaîne de caractères composée de n occurences du caractère c ;
- au moyen de la fonction de sélection d'une sous-chaîne `String.sub : string -> int -> int -> string` prenant en arguments une chaîne de caractères s, un indice de début i et une longueur l, et retournant le facteur de s de longueur l commençant à l'indice i ;
- au moyen de la fonction de concaténation de sous-chaînes `(^) : string -> string -> string` prenant en arguments deux chaînes de caractères et retournant leur concaténation, par exemple si `s1` vaut `"tata"` et `s2` vaut `"tutu"`, l'expression `s1^s2` s'évalue en `"tatatutu"`.

Accès.

- La fonction `String.length : string -> int` calcule la longueur d'une chaîne.
- Si `s` est une chaîne de longueur n et `i` un indice dans $[\![0, n[\![$, l'expression `s.[i]` vaut le caractère à la `i`-ème position dans `s`.

Modification. Il n'est **pas possible** de modifier une chaîne de caractères, au sens où il s'agit d'un tableau **non mutable** de caractères. Ainsi les expressions de la forme `s.[i] <- 'a'` ne sont pas syntaxiquement correcte. En revanche il est toujours possible de manipuler des références vers des `string`, ou des tableaux de caractères, ainsi on peut par exemple définir les fonctions[a] suivantes

```
let met_au_pluriel (sr:string ref) : unit =
  sr := (!sr)^"s"
```

```
let met_une_majuscule (c:char array) : unit =
  assert( Array.length c > 0);
  if ('a' <= c.(0) &&  c.(0) <= 'z')
  then let decalage = int_of_char 'A' -  int_of_char 'a' in
  c.(0) <- char_of_int ( int_of_char c.(0) + decalage )
  else ()
```

a. fonctions grossières qu'il est déconseillé de présenter aux prof de français...