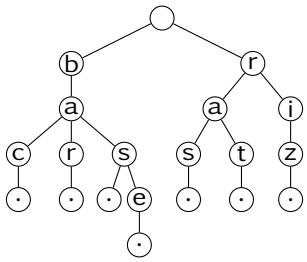
TP : Dictionnaire

Les mots d'un dictionnaire peuvent être représentés par un arbre en faisant en sorte que les préfixes communs à plusieurs mots apparaissent une seule fois :



Le dictionnaire ci-dessus représente la liste de mots : `bac, bar, bas, base, ras, rat, riz`

Question 1
Quel est l'intérêt de faire terminer chaque mot par un point ?

On introduit le type suivant pour représenter les arbres :

```
type dico = Nil | Noeud of char * dico * dico ;;
```

La première composante d'un **Noeud** contient le *caractère* associé à ce noeud, la seconde composante (*branche gauche*) permet d'accéder au *fils aîné* de ce noeud, et la troisième composante (*branche droite*) permet d'accéder au *frère cadet* de ce noeud.
On retrouve ainsi la correspondance entre les arbres binaires et les arbres d'arité quelconque.

Question 2
Donner le dictionnaire associé à l'arbre dessiné ci-dessus.
(Attention, ici fils gauche et droit ne jouent pas le même rôle)

Question 3
Rédigez en Caml *une fonction :*

```
chercher : string -> dico  -> bool
```

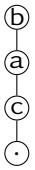
qui détermine si un mot appartient à un dictionnaire donné.

Question 4
Rédigez en Caml *une fonction :*

```
branche : string -> dico
```

qui prend pour argument un mot m et retourne pour résultat l'arbre réduit à la branche qui code le mot m.

Par exemple, au mot `bac` *sera associé l'arbre :*



Question 5
Rédigez en Caml *une fonction :*

```
inserer : string -> dico -> dico
```

qui prend pour arguments un mot m et un arbre a et qui retourne l'arbre auquel le mot m a été ajouté (sans ordre particulier)

Question 6
Rédigez en Caml *une fonction :*

```
creer : string list -> dico
```

qui prend pour argument une liste de mots et qui retourne le dictionnaire créé à l'aide de ces mots.

Question 7
Rédigez en Caml *une fonction :*

```
extraire : dico -> string list
```

qui renvoie la liste des mots contenus dans un dictionnaire donné.
(On pourra utiliser une fonction récursive auxiliaire ayant un accumulateur enregistrant le préfixe courant déjà lu)

Rappels sur *char* et *string*

— `String.length` donne la longueur d'une chaîne de caractères.

— Le caractère c est noté en Caml `'c'` (*l'apostrophe*)

— Si `s` est une chaîne de caractères (type *string*), alors les caractères composants `s` sont obtenus par : `s.[0]` , `s.[1]` , . . . , `s.[n-1]` (si n est la longueur de `s`).

— `String.sub : string -> int -> int -> string`
`( String.sub s i n )` permet de sélectionner la sous séquence de lettres de longueur n, contenant les lettres de s entre les indices i et $i + n - 1$.

— `s1 ^ s2` permet de concaténer les deux chaînes de caractères s_1 et s_2 en une seule chaîne.

— les fonctions `string_of_char` et `char_of_string` n'existent pas.
Cependant on peut les définir :

```
let string_of_char = String.make 1 ;;
let char_of_string s = s.[0];;
```

La fonction `string_of_char` transforme un caractère c en une chaîne de caractère de longueur 1 remplie du caractère c.
La fonction `char_of_string` renvoie le premier caractère de la chaîne de caractère en entrée.