le barème est provisoire (et sur 47 points) !

Ensemble $\mathcal{S}$ de chaînes de caractères.

`Question 1 : (2 )`
Définir la taille $\tau(\mathcal{S})$ d'un ensemble de chaînes de caractères.
Le nombre total N de caractères de l'ensemble des chaînes plus le nombre p de chaînes.
Mais comme il peut y avoir au plus 1 chaîne vide, $p \leq N + 1$, donc une taille en $\mathcal{O}(N)$.

Une structure de données pour représenter un ensemble de chaînes de caractères.

`Question 2 : (2 )`
À quelle condition la taille de la structure de données qui représente $\mathcal{S}$ est-elle dans $\mathcal{O}(\tau(\mathcal{S}))$?
Si toutes les feuilles de la structure sont `present` (blanches).

`Question 3 : (10)`
Écrire une fonction d'affichage de chaînes, que l'on appellera AS
- *qui prend en entrée $\mathcal{A}$, une structure de données comme vue ci dessus qui représente un ensemble $\mathcal{S}$ de chaînes de caractères*
- *et qui affiche les chaînes de $\mathcal{S}$ dans l'ordre lexicographique*

L'appel initial depuis `AS(A)` sera `AR(A,`V_D`)`

```
void AR(sommet* A, chaine Ch)

    si A==NULL alors retourner ;
    si A→present alors AC(Ch);
    AR(A→ptA,a(ch));
    AR(A→ptB,b(ch));
    retourner
```

`Question 4 : (2)`
Donner la complexité de votre fonction AS, la justifier et justifier qu'elle est optimum.
Le coup de la traversée de l'arborescence est dans $\mathcal{O}(\tau(\mathcal{S}))$
Quant au coût d'impression des chaînes de caractères il est dans $\Theta(\tau(\mathcal{S}))$ et on ne peut pas faire moins.
`Question 5 (1)`
Pourquoi l'algorithme qui consiste à trier d'abord les chaînes de la liste $\mathcal{L}$ dans l'ordre lexicographique, puis à les imprimer, n'est-il est pas de complexité linéaire ?
Il y aura à faire $\Theta(plog(p))$ opérations de comparaison chacune dans $\Theta(n)$. Le facteur $log(p)$ casse la linéarité.

`Question 6 : (15)`

Écrire la ou les méthodes qui vous seront nécessaires de manipulation de votre structure de données (à l'exception du constructeur) , et justifier pour chacune sa complexité en fonction des variables que vous choisirez vous même.

Il faut ecrire le constructeur par défaut $sommet :: sommet()$ $\{present = false;\ ptA = NULL,\ ptB = NULL;\}$
Il faut aussi écrire la méthode $sommet :: Inser(ch)$ qui insère une nouvelle chaîne ch dans un arbre :

si $CV?(ch)$ **alors**
  $present = true$
sinon
  si $a?(ch)$ **alors**
    si ptA **alors**
      $ptA \rightarrow Inser(CR(ch))$
    sinon
      $ptA = new\ sommet;$
      $ptA \rightarrow Inser(CR(ch))$
    fin
  sinon
    si ptB **alors**
      $ptB \rightarrow Inser(CR(ch))$
    sinon
      $ptB = new\ sommet;$
      $ptB \rightarrow Inser(CR(ch))$
    fin
  fin
fin
retourner *this*

Dont la complexité est trivialement dans $\Theta(|ch|)$.

`Question 7 : (15)`

Ecrivez maintenant l'algorithme `AL(`$\mathcal{L}$`)` *d'affichage de toutes les chaines de caractères de la liste* $\mathcal{L}$.

$sommet *\ \mathcal{A} = new\ sommet;$
$\mathcal{A} \rightarrow present = false;$
tant que $LN?(\mathcal{L})$ **faire**
  $ch = LP(\mathcal{L});\ \mathcal{L} = LR(\mathcal{L});$
  $\mathcal{A} \rightarrow Inser(ch)$
fin
$AS(\mathcal{A})$

La complexité amortie de la répétitive de construction de l'arbre est trivialement dans la taille de **la liste** de chaînes de caractères, et c'est la même que celle de l'affichage de la structure si toutes les listes sont différentes.