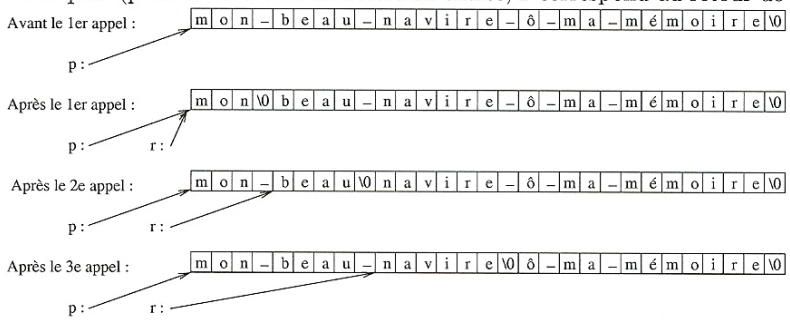
# Test d'informatique pour l'entrée en Master LI
## Durée : 1h.

1. On se propose d'écrire une fonction `strToken` qui, recevant une chaîne de caractères en paramètre, renvoie successivement tous les « mots[7] » contenus dans cette chaîne.
   Pour ce faire (cf exemple), la fonction va retourner à chaque appel sur une chaîne donnée (soit `p` ici) un pointeur sur la première lettre d'un mot dans `p`. Pour que cela fonctionne, il faut aussi que le premier séparateur après le mot pointé soit remplacé par un `'\0'`, de sorte que les fonctions habituelles de manipulation de chaînes de caractères puissent s'appliquer.
   (a) Écrire un algorithme qui, étant donné un pointeur n'importe où dans une chaîne, détermine la position du début et de la fin du prochain mot.
   (b) Déterminer quelle structure de donnée est nécessaire pour mémoriser l'ensemble des informations nécessaires pour que chaque appel de `strToken` renvoie effectivement ce que l'on souhaite.
   (c) Ecrire l'algorithme de la fonction `strToken`.
   (d) Comment pourrait-on envisager une version de `strToken` qui serait compatible avec une utilisation récursive ?

FIG. 3 – Exemple. (`p` est la chaîne de caractère en entrée, `r` correspond au retour de la fonction.)




2. On suppose que l'on dispose de fonctions/procédures permettant de manipuler une *pile* de caractères, dont les prototypes sont les suivants :[8]

   ```
   procedure empiler(x : char);
   function depiler : char;
   function vide : boolean;
   ```

   Ecrire un programme qui lit un texte, caractère par caractère, et vérifie que les parenthèses (`()`), les crochets (`[]`) et les accolades (`{}`) sont correctement équilibrés dans le texte.
3. [bonus] On se propose de trier dans l'ordre croissant un tableau contenant $N$ entiers. Proposer deux algorithmes pour réaliser le tri de ce tableau, en discutant succinctement les avantages et inconvénients de chaque algorithme.

[7] Les hypothèses habituelles (simples) s'appliquent : on suppose que tous les caractères qui ne sont pas des lettres servent de séparateur entre les mots ; on suppose que l'on dispose d'une fonction booléenne, `isletter`, qui renvoie vrai si son argument est un caractère alphabétique, et faux sinon.

[8] La procédure `empiler` empile son argument sur la pile, la fonction `depiler` renvoie le caractère sommet de pile, et le supprime, la fonction `vide` renvoie `true` si la pile ne contient aucun élément.