Exercice 6

L'ensemble de cet exercice est à traiter sur une feuille à part. NE RIEN INSCRIRE SUR CES FEUILLES, elles ne seront pas à rendre, au contraire vous les garderez à la fin de l'épreuve.

Un **langage de balisage** est un langage qui permet d'enrichir un texte avec des informations concernant sa mise en forme (principalement). Ces informations sont contenues dans des portions de texte spécifiques appelées **balises**, typiquement des portions de texte entre chevrons, comme `<gras>` ou `<fin_italique>`.

On appelle **texte brut** le texte sans mise en forme et incluant les balises, et au contraire, texte enrichi le texte mis en forme selon les balises. Dans le texte enrichi les balises n'apparaissent plus. Par exemple, le texte brut `<italique>Bonjour la <gras>MP2I<fin_gras> !!!<fin_italique>` correspondrait au texte enrichi *Bonjour la **MP2I** !!!*.

La langage HTML est sûrement l'exemple le plus connu de langage de balisage : il permet de définir la mise en forme d'une page web. Cependant, nous nous intéresserons à un langage de balisage beaucoup plus simple, inventé pour l'exercice, qui permet la mise en exergue (mise en valeur) de portions de texte. Dans ce langage,

- le texte initial ne contient aucun chevron, *i.e.* ni `<`, ni `>` ;
- toutes les balises seront entre chevrons (*i.e.* entre un chevron ouvrant et un fermant) ;
- une balise ne peut contenir elle-même une balise ;
- on distinguera les balises ouvrantes de la forme `<xxx>`, des balises fermantes de la forme `</yyy>`
- à chaque balise ouvrante `<xxx>` correspond une balise fermante `</xxx>` ;
- deux portions de texte encadrées par un couple de balises correspondantes ne peuvent se chevaucher à moins d'être imbriquées, bien sûr elles peuvent aussi être disjointes, autrement dit les balises doivent être placées dans le texte comme un bon parenthésage.

Détection de balises

Dans cette première partie, on s'intéresse déjà à la détection des balises, c'est-à-dire des portions de texte entre chevrons. En fait la première étape est de vérifier que les chevrons délimitent bien des portions de texte disjointes (le cas de portions de texte imbriquées est exclu puisqu'on a supposé qu'une balise ne peut contenir elle-même une balise).

Question 1 /4pt

Donner le code d'une fonction `ok_balises` qui vérifie que les chevrons de la chaîne de caractères passée en argument sont bien appariés et que les portions de chaîne qu'ils délimitent sont bien disjointes. Cette fonction doit en particulier satisfaire les tests ci-dessous. *On ne se préoccupe pas ici de savoir si les portions de textes entre chevrons définissent bien des balises, ni si les balises sont bien appariées. Notez qu'à chaque position d'une chaîne valide au sens de `ok_balises`, il n'y a que deux états possibles : soit on est dans une portion entre chevrons, soit on est hors des chevrons.*

```
assert(ok_balises("dvfjhgvdf<tt>"));
assert(ok_balises("dvfj</tt>hgvdf<tt>"));
assert(ok_balises("dvfj"));
assert(!ok_balises("<<tt>>"));
assert(!ok_balises("gjdhfqg > sdjbdvb"));
assert(!ok_balises("gjdhfqg < sdjbdvb"));
```

Dans certains cas on peut être amené à ignorer les balises. Par exemple si l'on veut étudier la fréquence d'apparition des mots sur une page web afin d'en déterminer le sujet, les mots contenus dans les balises sont peu pertinents, voir risqueraient de fausser le résultat. On doit donc pouvoir extraire d'un texte brut le texte initial, *i.e.* le texte où l'on a gommé les balises.

Question 2 /4pt
Donner le code d'une fonction `texte_init` qui prend en argument une chaîne de caractères `ch` valide selon `ok_balises` et retourne la chaîne correspondant au texte initial de `ch`. *Vous veillerez à préciser en commentaires les hypothèses et avertissements relatifs à cette fonction. On acceptera de réserver un espace mémoire potentiellement trop grand pour le résultat. On pourra aussi utiliser la fonction* `longueur` *qui retourne la longueur d'une chaîne de caractères en la parcourant, et donc en temps linéaire.*

Question 3 /1pt
Quel serait le gain, en termes d'efficacité et de complexité, de passer la longueur de la chaîne en deuxième argument à la fonction `texte_init`.

Texte bien balisé

Dans la suite de l'exercice, on suppose qu'on a seulement les trois paires de balises présentées dans le tableau ci-dessous. On remarque que les mises en formes associées à ces balises sont faites pour pouvoir être observées lors d'un affichage dans le terminal.

ouvrante	fermante	exemple de texte brut	texte enrichi associé
`<1>`	`</1>`	`...<1>ohé</1>...`	`...*ohé*...`
`<2>`	`</2>`	`...<2>ohé</2>...`	`...OHé...`
`<3>`	`</3>`	`...<3>matelot</3>...`	`...M-A-T-E-L-O-T...`

On remarque aussi que la mise en valeur induite par ces balises est graduelle : plus on veut mettre en valeur une portion de texte, plus on l'encadre par des balises en bas de ce tableau. On s'attend donc à avoir des balises de type `2` imbriquées dans des balises de type `1` mais pas le contraire. On ne demande pas de vérifier que cette contrainte est respectée, en revanche on limitera le degré d'imbrication des balises à 3, autrement dit une portion de texte brut peut se trouver entre trois paires de balises imbriquées mais pas plus. On donne quelques exemples de mises en formes pour les imbrications prévues.

exemple de texte brut	texte enrichi associé
`...<1>rendez-vous <2>demain</2></1> à 8h.`	`...*rendez-vous DEMAIN* à 8h.`
`...<1>rendez-vous <3>demain</3></1>...`	`...*rendez-vous D-E-M-A-I-N*...`
`<1>Rendez-vous <2>demain <3>matin</3></2></1>`	`*Rendez-vous DEMAIN M-A-T-I-N*`
`Surtout <2>apprenez votre <3>cours</3></2>`	`Surtout APPRENEZ VOTRE C-O-U-R-S`

On dira qu'un texte est **bien balisé** s'il est valide au sens de `ok_balises`, si toutes les balises sont parmi les 6 présentées ci-dessus, si chaque balise ouvrante du texte peut être associée à une balise fermante correspondante dans la suite du texte, de sorte que les portions de textes ainsi délimitées soient disjointes ou imbriquées, dans la limite de trois imbrications au plus.

La suite de cette partie consiste en des questions préalable au codage d'une fonction qui teste si une chaîne de caractères est bien balisée, le code de cette fonction étant long.

Afin de faciliter les comparaisons de balises en vue de vérifier leur correspondance, on représente chaque balise par un numéro. Ce numéro est un entier strictement positif pour une balise ouvrante, et un entier strictement négatif pour une balise fermante. Plus précisément le numéro d'une balise fermante est l'opposé de la balise ouvrante correspondante. Dans un soucis de simplicité, on choisit que le numéro de `<1>` est 1, celui de `<2>` est 2, et celui de `<3>` est 3. Les numéros de `</1>`, `</2>` et `</3>` s'en déduisent.

Question 4 /4pt
Donner le code d'une fonction `lit_balises` qui prend en argument une chaîne de caractères qui commence par une portion de texte entre chevrons, donc a priori par une balise, et qui retourne 0 si ce n'est pas l'une des balises susmentionnées (*i.e.* ni `<1>`, ni `<2>`, ni `<3>`, ni `</1>`, ni `</2>`, ni `</3>`), et qui retourne sinon le numéro associé à cette balise. *Par hypothèse, la chaîne en argument a au moins 2 caractères, `<` et `>`, mais on pourra supposer, pour simplifier, qu'elle en a au moins 4.*

Question 5 /1pt
Lorsqu'on applique `lit_balise`, combien de caractères de la chaîne passée en argument sont lus ? *On travaille toujours avec comme seules balises les trois couples de balises susmentionnés.*

Question 6 /1pt
Quelle structure de données abstraite paraît adaptée pour stocker les numéros de balises lors d'un parcours d'une chaîne de caractères en vue de déterminer si celle-ci est bien balisée ? Justifier succinctement.

Question 7 /1pt
Quelle implémentation concrète de cette structure de données paraît adaptée ? Justifier.

Question 8 /2.5pt
Lister les raisons pour lesquelles une chaîne valide selon `ok_balise` peut ne pas être bien balisée.

Question 9 /3.5pt
On suppose définie une fonction `bien_balise` qui teste si une chaîne de caractères valide au sens de `ok_balise` est bien balisée. Proposer un jeu de test *complet* pour cette fonction. *Si vous avez répondu à la question précédente vous pouvez indiquer pour les tests négatifs à la détection de quel défaut de la chaîne ils correspondent.*

Mise en forme

On envisage dans cette partie de générer le texte enrichi à partir d'un texte brut bien balisé. On se limite à des questions préalables, la fonction réalisant la mise en forme étant assez longue.

Question 10 /2pt
Que dire de la longueur l' du texte enrichi correspondant à un texte brut de longueur l. A-t-on toujours $l' \geqslant l$, ou toujours $l' \leqslant l$, ou bien cela dépend-il ? Justifier par une démonstration ou des contre-exemples.

Question 11 /2pt
Donner le code d'une fonction `majuscule` qui prend en argument un caractère, et qui retourne la lettre majuscule correspondante si ce caractère est une lettre minuscule, et le caractère lui même sinon. *On rappelle que dans le code ASCII utilisé pour encoder les caractères en C, toutes les lettres minuscules (resp. majuscules) sont consécutives et dans l'ordre alphabétique donc entre `'a'` et `'z'` (resp. entre `'A'` et `'Z'`).*