Introduction à la Programmation 1

TP Noté

Université Paris-Diderot

Instructions:

- Vous avez le droit d'utiliser les fonctions de la bibliothèque standard vues en cours et en TP. Par exemple : "`s.length()`" ou "`s.charAt(i)`" (où `i` est le nom d'une variable contenant un nombre entier).
- Le TP est à faire sur les ordinateurs de la salle, sur n'importe quel éditeur de texte.
- Le TP est à rendre par mail dès la fin de l'heure à alessio.mansutti@lsv.fr. Le mail devra avoir pour sujet "TP noté - `$NOM $PRENOM`" (où `$NOM` et `$PRENOM` doivent être remplacés par votre nom et prénom).
- Rapellez-vous de de tester vos solutions en utilisant la procédure

```
public static void main(String[] args)
```

- C'est toujours mieux de lire complètement un exercice avant de commencer à le résoudre.

Exercice 1 (Ton premier algorithme de tri)

Dans cet exercice, on va écrire une procédure `sortBin` *qui prend en paramètre une chaîne de caractères* `s` *qui ne peut contenir que des zéros ou des un, et affiche la version triée de* `s`*, c'est-à-dire la chaîne de caractères qui (1) peut être obtenu à partir de* `s` *en permutant ses lettres et (2) où tous les zéros précèdent tous les un. Par exemple,* `sortBin("010010")` *affiche sur le terminal* `000011`*.*

1. *On commence par écrire une fonction* `isOne` *qui prend en argument un caractère* `ch` *et qui renvoie* `true` *si* `ch` *c'est égal à* `'1'` *et* `false` *sinon.*
2. *Écrire une fonction* `howManyOnes` *qui prend en argument une chaîne de caractères* `s` *et calcule combien de fois le caractère* `'1'` *apparaît dans* `s`*. Par exemple,* `howManyOnes("01001100")` *renvoie* `3`*. Si possible, utiliser la fonction* `isOne` *pour écrire cette fonction.*
3. *Écrire une fonction* `howManyZeros` *qui prend en argument une chaîne de caractères* `s` *qui ne peut contenir que de zéros ou des un, et calcule combien de fois le charactére* `'0'` *apparaît dans* `s`*. Par exemple,* `howManyOnes("01001100")` *renvoie* `5`*. Utiliser la fonction* `howManyOnes` *pour écrire cette fonction plus facilement.*
4. *Écrire une procédure* `sortBin` *qui prend en argument une chaîne de charactéres* `s` *qui ne contient que de zéros ou des un, et affiche sa version triée. Un deuxième exemple :* `sortBin("1110010")` *affiche* `0001111`*. Utiliser les fonctions* `howManyZeros` *et* `howManyOnes` *pour écrire cette procédure.*

□

Exercice 2 (Fermez les parenthèses !)

Tous les codeurs savent que chaque parenthèse ouvrante `'('` *a besoin d'une parenthèse fermante* `')'`*, et vice versa. Dans cet exercice, on va écrire une fonction* `wellFormed` *qui prend en argument une chaîne de caractères* `s` *qui (pour simplifier) ne contient que des* `'('` *ou des* `')'`*. La fonction renvoie* `true` *si, dans* `s`*, il est possible associer à chaque parenthèse* `'('` *exactement une parenthèse* `')'` *qui apparaît a sa droite, et en plus si le nombre des parenthèse* `'('` *et* `')'` *est le même. Autrement, la fonction renvoie* `false`*.*

1. *On commence par écrire une fonction* `parseChar` *qui prend en argument un caractère* `c` *et retourne* `1` *si* `c` *est égal à* `'('`*, et retourne* `-1` *si* `c` *est égal à* `')'`*. Autrement, elle renvoie* `0`*.*

On va maintenant implémenter la fonction `wellFormed` *décrite au début de l'exercice, de la façon suivante :*

— *La fonction prend en argument une chaîne de caractères* `s` *qui ne peut contenir que des* `'('` *ou des* `')'`.
— *On définit une variable locale entière* `ouvertes`. *Lors du parcours du mot,* `ouvertes` *augmente de* `1` *à chaque parenthèse ouvrante et diminue de* `1` *pour chaque parenthèse fermante (utiliser ici la fonction* `parseChar`*).*
— *Si* `ouvertes` *devient negative apres sa actualisation, la fonction renvoie immédiatement* `false`.
— *Après avoir terminé l'analyse des les caractères dans* `s`, *la fonction renvoie* `true` *si* `ouvertes` *est égal à* `0`, *et* `false` *sinon.*

Testez la fonction sur les entrées suivantes : `"(()())"`, `"(()()"` *et* `"(()))("`. *La fonction dois renvoier* `true` *seulement sur la première entrée.* □