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 (Le facteur le plus significatif)

Dans cet exercice, on va écrire une procédure `largestFactor` *qui prend en paramètre un nombre* `n` *et qui affiche le plus grand facteur propre de* `n`*, c'est-à-dire le plus grand diviseur de* `n` *qui est strictement inférieur à* `n`*. Par exemple,* `largestFactor(12)` *affiche* `6`*.*

1. *Écrire une fonction* `divides` *qui prend en argument deux nombres* `n` *et* `i` *et qui renvoie* `true` *si* `i` *est un diviseur de* `n` *et* `false` *sinon.*
2. *Écrire une procédure* `inverseOrder` *qui prend en argument un nombre* `n` *et qui affiche tous le nombres entre* `n-1` *et* `2` *(en ordre inverse). Par exemple,* `inverseOrder(5)` *affiche sur le terminal* `432`*.*
3. *Écrire la procédure* `largestFactor` *qui prend en paramètre un nombre* `n` *et qui affiche le plus grand facteur propre de* `n`*. Un deuxième exemple :* `largestFactor(112)` *affiche* `56`*. Pour écrire cette procédure, utiliser la fonction* `divides`*, ainsi que l'idée utilisée pour la boucle* ***for*** *de la procédure* `inverseOrder`*.*

□

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.* □