Sélection internationale Session 2015
École Normale Supérieure Paris
Épreuve de culture scientifique - **Informatique** Durée : 2 hours

Pour les candidats ayant choisi **l'informatique** comme **spécialité secondaire**

Si vous ne parvenez pas à répondre à une question, vous pouvez cependant l'utiliser comme hypothèse pour les questions suivantes.

L'usage de la calculatrice n'est pas autorisé.

Exercice 1.
Une expression arithmétique simple est composé des caractères `1`, `+` et des parenthèses. Presque tout entier peut être représenté par une expression arithmétique simple. Par exemple, les expressions arithmétiques simples suitvantes représentent l'entier 14:

```
1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1
((1+1) (1+1+1+1+1))+((1+1) (1+1))
(1+1) (1+1+1+1+1+1+1)
(1+1) (((1+1+1) (1+1))+1)
```

Décrire un algorithme pour calculer, étant donné un entier n, le nombre minimum de 1 qu'il faut dans une expression arithmétique simple dont la valeur est n. Le nombre de parenthèse n'est pas important, seulement le nombre de 1. Par exemple, pour $n = 14$, votre algorithme devrait retourner 8.

Votre algorithme devrait calculer en temps polynomial en n.

Exercice 2.
Pour cet exercice, tous les tableaux sont des tableaux d'entiers et un tableau *trié* est un tableau où les éléments sont en ordre croissant.

1. Décrire un algorithm $O(n)$ qui
   - prend en entrée un tableau A et un entire i tel que $A[1..i]$ est trié et $A[i+1..n]$ est trié, et
   - modifie A tel que A devient trié.

   Votre algorithm ne doit pas créer de nouveaux tableaux. (En d'autre mots, votre algorithme devrait être *en place.*)

Le *tri fusion* est un algorithme de tri qui prends un tableau non-trié et applique votre algorithme de la question 1 de manière récursive.

Dans cette question, nous allons examiner une variante du tri fusion où avant de fusioner, au lieu de deux sous-tableaux triés, le tableau peut être divisé en plusieurs sous-tableaux triés. Une suite représent la longueur de ces sous-tableaux triés.

Nous voulons mettre à jours cette suite pour reflèter le changement quand un nouveau sous-tableau est ajouté et le tri fusion est appliqué sur des pairs de sous-tableaux. Pour le reste de cette question, nous allons seulement regarder cette suite de longueurs (au lieu des détails du tri du tableau).

Considérons une fonction de mise-à-jours ACTUALISER sur une suite L d'entiere positifs non-nulle (représentant des longueurs de sous-tableau).

```
fonction ACTUALISER(L, valeur)
  AJOUTER(L, valeur)
  répéter indéfiniment
    n ← LONGUEUR(L)
    si n > 2 et L[n − 2] ≤ L[n − 1] + L[n] et L[n − 2] < L[n] alors
      FUSION(L, n − 2, n − 1)
    sinon si n > 2 et L[n − 2] ≤ L[n − 1] + L[n] et L[n − 2] ≥ L[n] alors
      FUSION(L, n − 1, n)
    sinon si n ≥ 2 et L[n − 1] ≤ L[n] alors
      FUSION(L, n − 1, n)
    sinon
      return
```

Le fonction ci-dessus utiliser les sous-programmes suivants.

LONGUEUR(L) retourne le nombre d'éléments actuellement dans L.

AJOUTER(L, valeur) ajoute valeur à la fin de la suite L.

FUSION$(L, i, i+1)$ remplace les éléments aux indices i et $i+1$ par un seul élément (à l'indice i) dont la valeur est leur somme. Par exemple FUSION$(L, 3, 4)$ quand L est

$$0, 2, 4, 6, 8$$

donne

$$0, 2, 10, 8$$

puisque $4 + 6 = 10$.

$L[i]$ retourne le $i$ème élément de la suite. $L[n]$ est le dernier élément de la suite et $L[1]$ est le premier élément.

2. Décrire une structure de donnée qui permet
   - les opérations LONGUEUR and AJOUTER en $O(1)$ et
   - l'opération FUSION en $O(1)$ quand nous fusionnons les deux deniers éléments ou les deux avant-derniers éléments. I.e., les seuls entrées permis à FUSION$(L, i, i+1)$ sont $i =$LONGUEUR$(L) - 1$ et $i =$LONGUEUR$(L) - 2$.

3. Montrer qu'en partant d'une suite vide, en votre structure de donnée, appliquer ACTUALISER k fois se fait en temps $O(k)$.

4. Quand la fonction actualiser retourne, les inégalités suivantes sont vraies (si L a encore suffisament d'éléments):
   - $L[n-1] > L[n]$
   - $L[n-2] > L[n-1] + L[n]$

   Montrer que si
   - $L[i-1] > L[i]$
   - $L[i-2] > L[i-1] + L[i]$

   pour tout i alors le premier élément de L a une valeur exponentielle en la longueur de la suite L.

5. Montrer qu'en partant d'une suite vide, il existe un ensemble d'appels à ACTUALISER (et aucune autre fonction) tel qu'il est faux que

   - $L[i-1] > L[i]$
   - $L[i-2] > L[i-1] + L[i]$

   pour tout i à la fin de ces appels.

6. Montrer que si on modifie ACTUALISER en ACTUALISERMODIFIÉ tel que

   - $L[i-1] > L[i]$
   - $L[i-2] > L[i-1] + L[i]$

   est vrai pour $i = n$ et $i = n-1$ à la fin de chaque appel à ACTUALISERMODIFIÉ, alors en partant d'une suite vide, après n'importe quel nombre d'appels à ACTUALISERMODIFIÉ (et aucune autre fonction)

   - $L[i-1] > L[i]$
   - $L[i-2] > L[i-1] + L[i]$

   est vrai pour tout i.