Algorithmique
Contrôle nº 1 (C1)

Info-spé (s3)
Epita

D.S. 310844.97 BW (5 Nov. 2013 - 10 :00)

Consignes (à lire) :

- ☐ Vous devez répondre sur **les feuilles de réponses prévues à cet effet**.
  - Aucune autre feuille ne sera ramassée (gardez vos brouillons pour vous).
  - Répondez dans les espaces prévus, **les réponses en dehors ne seront pas corrigées** : utilisez des brouillons !
  - Ne séparez pas les feuilles à moins de pouvoir les ré-agrafer pour les rendre.
  - Aucune réponse au crayon de papier ne sera corrigée.
- ☐ La présentation est notée en moins, c'est à dire que vous êtes noté sur 20 et que les points de présentation (2 au maximum) sont retirés de cette note.
- ☐ **Les algorithmes :**
  - Tout algorithme doit être écrit dans le langage Algo (pas de C, Caml ou autre).
  - Tout code Algo non indenté ne sera pas corrigé.
  - Tout ce dont vous avez besoin (types, routines) est indiqué en **annexe** (dernière page) !
- ☐ Durée : 2h00

Exercice 1 (Hachage linéaire – *2 points*)

Supposons l'ensemble de clés suivant E={data, kirk, neelix, odo, picard, q, quark, sisko, tuvok, worf} ainsi que la table 1 des valeurs de hachage associées à chaque clé de cet ensemble E. Ces valeurs sont comprises entre 0 et 10 ($m = 11$).

TABLE 1 – Valeurs de hachage

data	4
kirk	5
neelix	3
odo	1
picard	7
q	6
quark	2
sisko	7
tuvok	1
worf	7

Représenter la gestion des collisions pour l'ajout de toutes les clés de l'ensemble E dans l'ordre de la table 1 (de `data` jusqu'à `worf`) et dans le cas du hachage linéaire avec un coefficient de décalage $d = 4$.

Exercice 2 (Hachage : Questions... – *3 points*)

1. Citez trois propriétés que doit posséder une fonction de hachage.
2. A quoi doit-on une collision secondaire ?
3. Collisions mises à part, quel phénomène provoque le hachage linéaire et qu'envisage t-on pour le résoudre ?

Exercice 3 (Hachage : Tableaux valides – *3 points*)

Supposons les clés de A à G avec les valeurs de hachage données dans la table 2.

TABLE 2 – Valeurs de hachage

clés	A	B	C	D	E	F	G
h(clés) m=7	2	0	0	4	4	4	2

Si celles-ci sont insérées dans un ordre quelconque, selon le principe du hachage linéaire (avec $d = 1$), dans un tableau initialement vide de taille 7, quels tableaux parmi les suivants ne peuvent pas résulter de l'insertion de ces clés ?

TABLE 3 – Tableaux possibles ?

	0	1	2	3	4	5	6
Tableau (A)	C	G	B	A	D	E	F
Tableau (B)	F	G	B	D	A	C	E
Tableau (C)	B	C	A	G	E	D	F
Tableau (D)	G	E	C	A	D	B	F

Exercice 4 (Des croissants – *5 points*)

Écrire un algorithme qui affiche les clés contenues dans un arbre 2-3-4 en ordre décroissant. On supposera l'arbre non vide au départ. L'optimisation sera (comme d'habitude) prise en compte.

Exercice 5 (Arbre 2-3-4 : insertion – *7 points*)

Rappel : insertion d'un élément avec principe de précaution

Le principe donné ici est celui de l'insertion d'une clé x dans un arbre A non vide, dont la racine n'est pas un 4-nœud (ces cas sont gérés à part) utilisant uniquement l'éclatement (pas de rotations).

On commence par rechercher le "point d'insertion" de la clé x dans la liste des clés de la racine. Ensuite, si la clé n'est pas présente, il reste deux possibilités :
- On est en feuille : il ne reste plus qu'à insérer la clé.
- En nœud interne : avant de descendre sur le bon fils, il faut s'assurer que sa racine n'est pas un 4-nœud, et l'éclater si elle l'est (sauf si la clé à insérer est déjà présente...).

Compléter la fonction récursive `insert_234` qui insère un nouvel élément dans un arbre 2-3-4 sauf si celui-ci est déjà présent. Votre fonction doit obligatoirement utiliser le principe donné.

▷ On suppose implémentée la procédure d'éclatement dont les spécifications sont les suivantes :
La procédure `eclate (A, i)` éclate le fils n°i de l'arbre A.
- L'arbre A existe et sa racine n'est pas un 4-nœud.
- Le fils n° i de A existe et sa racine est un 4-nœud.

▷ La fonction `insert_234` sera appelée par la fonction suivante :

```
algorithme fonction insertion_a234 : booleen
    parametres locaux
        t_element    x
    parametres globaux
        t_a234       A

    variables
        entier     i
        t_a234     T
debut
    si A = NUL alors
        allouer(A)
        A↑.nbcles ← 1
        A↑.cle[1] ← x
        A↑.fils[1] ← NUL
        A↑.fils[2] ← NUL
        retourne vrai
    sinon
        si A↑.nbcles = 3 alors
            allouer (T)
            T↑.nbcles ← 0
            T↑.fils[1] ← A
            A ← T
            eclate (A, 1)
        fin si
        retourne insert_234 (x, A)
    fin si
fin algorithme fonction insertion_a234
```

Annexes

Type de données représentant les arbres 2-3-4 :

```
constantes
    degre = 2
types
     /* déclaration du type t_ element */
    t_a234    = ↑ t_noeud_234
    tab3cles  = (2*degre-1) t_element
    tab4fils  = (2*degre) t_a234
    t_noeud_234 = enregistrement
        entier    nbcles
        tab3cles  cle
        tab4fils  fils
    fin enregistrement t_noeud_234
```

Rappel : dans le vecteur des fils, les k premiers fils sont à NUL pour les k-nœuds externes.

Routines autorisées

En dehors d'indications dans l'énoncé les routines autorisées sont :

Allocation mémoire

- `allouer (`p`)` : alloue de la mémoire pour une nouvelle variable et place son adresse dans p.
- `liberer (`p`)` : libère la mémoire occupée par la variable "pointée" par p.

Affichage

- `ecrire (`a`, `b` ...)` : affiche successivement les valeurs a, b ....