DS2 - Algorithmes et Structures de Données
Mardi 12 Janvier 2016

Durée 3H – Documents non autorisés

1. Jeu de dominos - (5 pts)

Un Domino est une paire qui contient 2 chiffres (de 1 à 6), par exemple 1:6.
Une chaine de dominos forme une suite, si les chiffres voisins coïncident sur chaque paire des dominos consécutifs, par exemple : 1:6 6:4 4:4 4:2 2:3
Soit une chaîne de dominos. On souhaite calculer la position `i` du début de la plus longue suite dans la chaîne et aussi calculer la taille `l` de cette suite.

Les algorithmes que vous écrirez devront être optimaux.

1.1. Ecrire une **procédure** `calculer-suite (E t : tab-dom, n : entier ; S i,l : entier)`
où la chaîne de dominos est implémentée par un tableau de dominos `t` (`n` étant le nombre de dominos dans le tableau).

```
Const max = 100
Type  domino = Enregistrement
                  gauche, droite : entier
               FinEnregistrement
      tab-dom = tableau[1..max] de domino
```

1.2. Ecrire une **procédure** `calculer-suite (E ld : liste-dom ; S i,l : entier)`
où la chaîne de dominos est implémentée par une liste chaînée `ld` (par pointeurs) de dominos.

```
Type  domino = Enregistrement
                  gauche, droite : entier
                  suiv : ^domino
               FinEnregistrement
      liste-dom = ^domino
```

2. Fichier texte - (3 pts)

Ecrire une procédure qui modifie un fichier texte en remplaçant toutes les occurrences d'une chaine de caractères (c1) par une autre (c2).

3. Quickselect - (4 pts)

Quickselect est un algorithme de sélection qui retourne le $k^{ème}$ plus petit élément `e` dans une séquence d'éléments non ordonnée. Il utilise la même approche que quicksort, choisissant un élément à la fois, afin de partitionner les éléments selon le pivot. Cependant, au lieu d'appliquer la récursion sur les deux parties, l'algorithme quickselect ne l'applique que sur le côté contenant l'élément qu'il cherche.
Ecrire la **procédure récursive** `quick-select(E/S t : tab-entier, E k, inf, sup : entier, S e : entier)`
qui effectue la recherche du $k^{ème}$ plus petit entier `e` dans le tableau d'entiers `t` entre les bornes `inf` et `sup`.

4. Généalogie - (8 pts)

On souhaite représenter la généalogie d'une famille par la structure de données suivante :

```
Type liste-pers = ^personne
     personne =   Enregistrement
                        nom : string
                        prénom : string
                        date-nais : string
                        sexe : string
                        pere : ^personne
                        mere : ^personne
                        couple-actuel : ^personne
                        enfants : tableau [1..10] de ^personne
                  FinEnregistrement
Var lp : liste-pers
```

La liste `lp` **est triée dans l'ordre alphabétique** sur les champs `nom` puis `prénom`. On considère qu'il n'y a pas deux personnes avec les mêmes nom et prénom. On considère aussi qu'une personne n'a pas plus de 10 enfants.

Vous ferez un dessin et expliquerez la méthode adoptée pour chaque procédure ou fonction écrite.

4.1. **Faire un dessin** de la structure de données sur 3 générations : grands-parents (2 grands-mères et 2 grands-pères), parents et enfants. On mettra à nil les parents des grands-parents.
4.2. Ecrire une **procédure** qui ajoute un enfant de père et mère donnés par leur nom et prénom (personnes présentes dans la liste `lp`), sans enfant, ni en couple actuellement.
4.3. Ecrire une **fonction** qui renvoie un pointeur sur le parent commun si 2 personnes sont demi-frères/demi-sœurs, nil sinon.
4.4. Ecrire une **fonction** qui renvoie un pointeur sur le parent d'affiliation (père ou mère) si 2 personnes sont cousins germains, nil sinon.
4.5. Ecrire une **procédure** qui écrit à l'écran le prénom et le nom de toutes les personnes qui sont séparées (être parent et ne plus être en couple avec le père ou la mère des enfants).