DS2 - Algorithmes et Structures de Données
Jeudi 17 Janvier 2019

Durée 3H – Documents non autorisés

1. Tableaux - 5 pts

Soit un tableau de contingence qui répartit 592 personnes en fonction de la couleur de leurs yeux et de leurs cheveux :

	cbrun	cchatain	croux	cblond
ymarron	68	119	26	7
ynoisette	15	54	14	10
yvert	5	29	14	16
ybleu	20	84	17	94

Le tableau de profils lignes (ou colonnes) est calculé en divisant les effectifs par les sommes en ligne (ou en colonne). Ex : 68 / 220 = 0.31 (31% des yeux marrons ont les cheveux bruns) et 68 / 108 = 0.63 (63% des cheveux bruns ont les yeux marrons).

Profils-lignes (effectifs divisés par la somme en ligne)

	cbrun	cchatain	croux	cblond
ymarron	0.31	0.54	0.12	0.03
ynoisette	0.16	0.58	0.15	0.11
yvert	0.08	0.45	0.22	0.25
ybleu	0.09	0.39	0.08	0.44

Profils-colonnes (effectifs divisés par la somme en colonne)

	cbrun	cchatain	croux	cblond
ymarron	0.63	0.42	0.37	0.06
ynoisette	0.14	0.19	0.20	0.08
yvert	0.05	0.10	0.20	0.13
ybleu	0.19	0.29	0.24	0.74

```
Const n = 4 {nombre de couleurs d'yeux}
      m = 4 {nombre de couleurs de cheveux}
Type mat = tableau[1..n,1..m] de réel
```

1.1. Ecrire une fonction `profils-lignes` qui calcule le tableau profils-lignes.

```
Fonction profils-lignes (m : mat, n,m : entier) : mat
```

1.2. Ecrire une fonction `profils-colonnes` qui calcule le tableau profils-colonnes.

```
Fonction profils-colonnes (m : mat, n,m : entier) : mat
```

2. Arbre binaire - 5 pts

2.1. Un **arbre binaire entier** est un arbre dont tous les nœuds possèdent zéro ou deux fils.
Ecrire en pseudo-langage la fonction `est-entier` en utilisant les opérations du TAD `arbrebin` :

```
Fonction est-entier (a : arbrebinaire) : booléen
```

2.2. Un **arbre binaire parfait** est un arbre binaire entier dans lequel toutes les feuilles sont à la même profondeur.
Ecrire en pseudo-langage la fonction `est-parfait` en utilisant les opérations du TAD `arbrebin` :

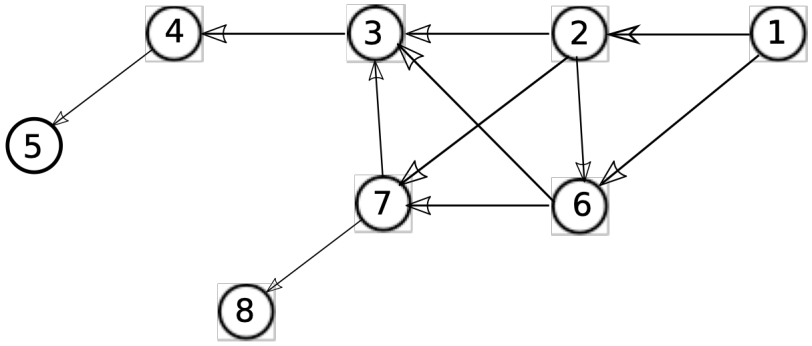
```
Fonction est-parfait (a : arbrebinaire) : booléen
```

3. Pointeurs – 10 pts

On peut représenter un graphe orienté par sa matrice d'adjacence mais aussi par la liste chaînée des nœuds qui pointent chacun sur la liste de ses fils (appelée **représentation par pointeurs**). On utilisera les structures de données suivantes :

```
Const max = 8 {nombre de nœuds du graphe orienté}
Type mat-adj = tableau [1..max, 1..max] de booléen
     graphe = ^nœud
     nœud = Enregistrement
                num : entier  {non nul}
                pfils : ^fils
                nsuiv : ^nœud
            FinEnregistrement
     fils = Enregistrement
                pnœud : ^nœud
                fsuiv : ^fils
            FinEnregistrement
```

Exemple de graphe orienté avec sa matrice d'adjacence :



$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

3.1. Dessiner la représentation par pointeurs correspondant au graphe orienté donné en exemple.

3.2. Ecrire en pseudo-langage une procédure qui ajoute à la représentation par pointeurs d'un graphe orienté un nœud donné par son numéro (`num`) qui n'est relié à personne. **Faire un dessin**.

```
Procédure ajouter-nœud (E num : entier ; E/S g : graphe)
```

3.3. Ecrire en pseudo-langage une procédure qui ajoute à la représentation par pointeurs d'un graphe un arc orienté de `num1` vers `num2`. **Faire un dessin**.

```
Procédure ajouter-arc (E num1,num2 : entier ; E/S g : graphe)
```

3.4. Ecrire en pseudo-langage une procédure qui supprime de la représentation par pointeurs d'un graphe orienté un nœud donné par son numéro (champ `num`). Penser à supprimer le nœud de la liste des fils de ses parents. **Faire un dessin**.

```
Procédure supprimer-nœud (E num : entier, E/S g : graphe)
```

3.5. Ecrire en pseudo-langage une procédure qui permet de construire la représentation par pointeurs d'un graphe orienté donné par sa matrice d'adjacence.

```
Procédure créer-graphe (E m : mat-adj, n : entier ; S g : graphe)
```

3.6. Ecrire en pseudo-langage une procédure qui supprime de la représentation par pointeurs d'un graphe orienté toutes ses feuilles (qui n'ont pas de fils, suppression des nœuds 5 et 8 sur l'exemple). Attention, la suppression d'une feuille peut faire apparaître de nouvelles feuilles qu'il faut de nouveau supprimer (nœud 4 sur l'exemple).

```
Procédure supprimer-feuilles (E/S g : graphe)
```