Contrôle structures de données en C

1 heure, documents non autorisés

novembre 2012

1 Listes chaînées

Q1. Les polynômes étant d'usage assez courant, on veut les représenter par un type abstrait de données *Polynome*. Un polynôme de degré n est de la forme $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_i x^i + \cdots + a_1 x^1 + a_0$ où les a_i représentent les coefficients et les i représentent les degrés. Chaque $a_i x^i$ est appelé *monome*.

Un monôme pourra être représenté par une structure composée des éléments suivants : le coefficient, le dégré et un pointeur sur le monôme suivant. Parmi ces trois propositions, laquelle représente le type *monome* et le type abstrait de données *Polynome* :

Proposition 1 :
```
typedef struct monome {
  int *coefficient;
  int *degre;
  struct monome *suiv;
}lmonome;
typedef lmonome *Polynome;
```

Proposition 2 :
```
typedef struct poly {
  int coefficient;
  int degre;
  struct monome **suiv;
}lmonome;
typedef struct monome  *Polynome;
```

Proposition 3 :
```
typedef struct monome {
  int coefficient;
  int degre;
  struct monome *suiv;
}lmonome;
typedef lmonome  *Polynome;
```

Q2. Dans la proposition retenue quelle est, en octets, la taille nécessaire pour représenter *un monome* en mémoire.

En utilisant la proposition retenue, dessiner une variable `Polynome poly` pointant la liste chaînée représentant en mémoire le polynôme $4x^3 + 3x^2 + 1$.

Q3. écrire une fonction C `createMonome` qui crée *un monome* et le retourne comme résultat.

```
Polynome createMonome(int coeff, int degre) {
```

De la même façon, écrire une procédure C `insererMonome` qui insére un `nouveau` monome entre un monome `existant` et son monome suivant.

```
void insererMonome(polynome existant, polynome nouveau) {
```

On part d'un polynôme $3x^2 + 1$ représenté en mémoire au moyen d'une liste chaînée pointée par la variable `Polynome poly`. On utilise les primitives de manipulation de la façon suivante : `insererMonome(poly, createMonome(2, 1))`. Dessiner la liste chaînée dont on part et celle obtenue après utilisation des primitives.

Avant :

Après :

Q4. écrire une fonction `AffichePolynome` qui affiche les coefficients et les dégrés d'un polynome.

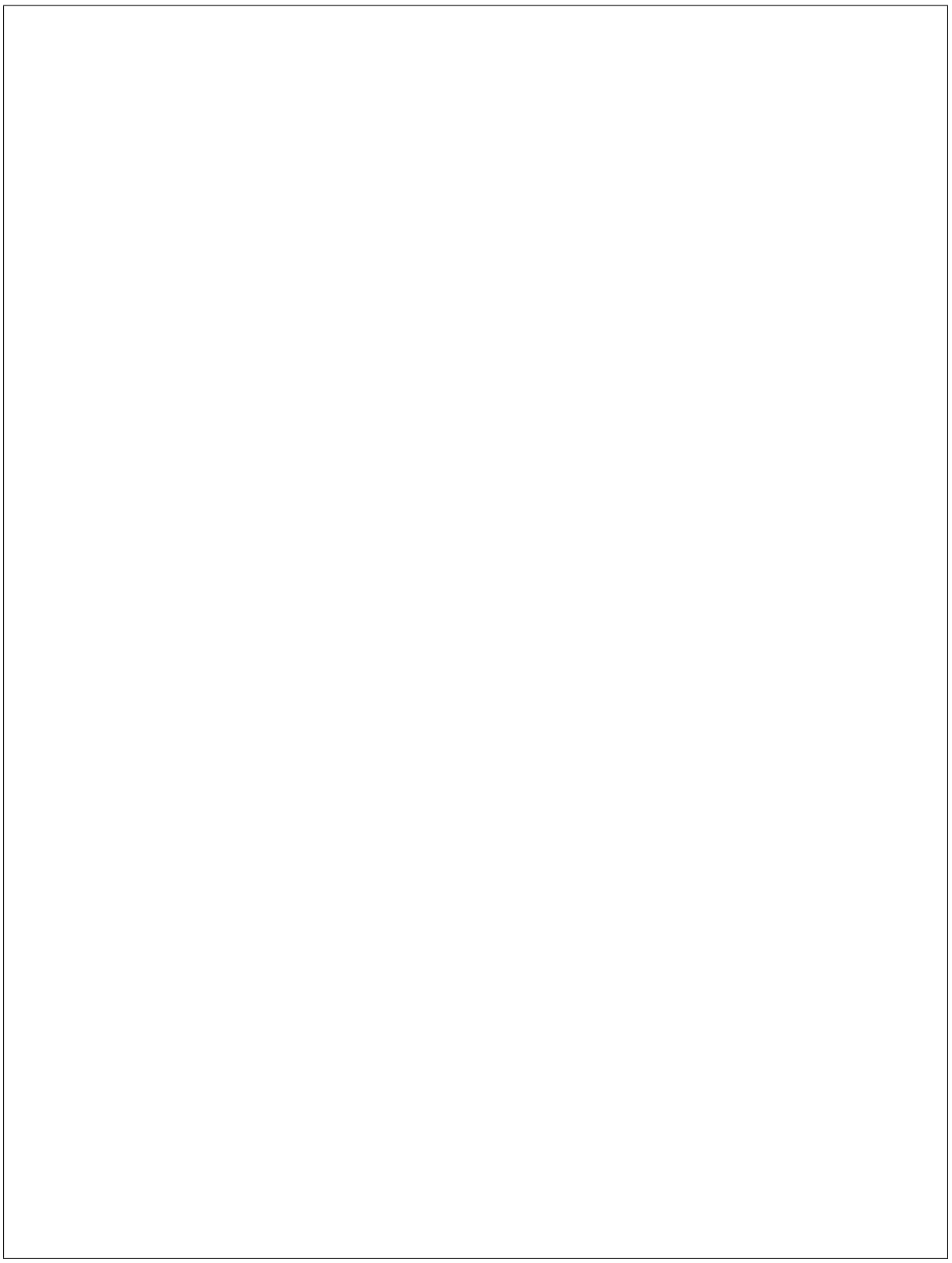
```
void AffichePolynome(Polynome poly) {
```

Q5. écrire une version récursive de l'affichage des coefficients et des dégrés d'un polynome `AfficheRecurPolynome` .

```
void AfficheRecurPolynome(Polynome poly) {
```

Arbres

Q6. Donner, en expliquant les différentes étapes, l'arbre équilibré contenant les nombres suivants : 57, 42, 17, 32, 12, 7, 24, 37, 62, 22, 47, 52.



Q7. Donner **l'arbre équilibré** après la suppression du noeud 57 de l'arbre construit dans la question précédente.