TD 2
Polynomes

TAD
Semestre 2

1

2 spécifications fonctionnelles du tad Liste[T]

```
1  -- determine la longueur de la liste l
2  fonction longueur (entree l <Liste[T]>
3    retourne <Entier>;
4
5  -- renseigne l'element i de la liste l
6  --necessite l <= i <= longueur (l)
7  fonction ieme (entree l <Liste[T]>, entree i <Entier>)
8    retourne <T>
9  declenche listeVide, rangInvalide;
10
11 --insere un element dans la liste l
12 -- necessite i <= i <= longueur(l)+1
13 procedure inserer (maj l <Liste[T], entree i <Entier>, entree e<T>)
14   declenche rangInvalide, listePleine;
15
16 -- retire l'element d'un rang i de la liste l
17 -- necessite 1<= i <= longueur de l
18 procedure supprimer (maj l <Liste[T]>, entree i <Entier>)
19   declenche rangInvalide, listeVide;
20
21 -- construit une liste vide
22 procedure creerListe (sortie l <Liste[T]>);
```

3 Algorithme d'une liste

```
1  procedure parcourirListe(entree l <Liste[Entier])
2  glossaire
3    i <Entier>;
4  debut
5    i <- 1;
6    tantque i <= longueur(l)faire
7      traiterElement(ieme, l, i);
8      i <- i + 1 ;
9    fin tantque;
10 fin
```

Listing 1 – Opérations en tant que client

4 Utilisation du Type Abstrait de Données Liste[T]

$$\begin{aligned} P(x) &= 3 + 4x + 2x^3 - 4x^5 + 7x^8 - x^9 + x^{10} \\ Q(x) &= 5 + 2x + 3x^4 - 4x^5 + 2x^6 \end{aligned}$$

$$P(x) + Q(x) = 8 + 6x + 2x^3 + 3x^4 + 2x^6 + 7x^8 - x^9 + x^{10}$$

```
creer liste vide;
se positionner surle premier element de P(x) et Q(x);
tantque il reste un monome dans P(x) et Q(x) faire
  si les puissances sont egales alors
    construire la somme des deux monomes;
    si coefficient de la somme different de zero alors
      inserer somme dans liste resultat;
    fin si;
    passer au monome suivant de P;
    passer au monome suivant de Q;
  sinon
    si puissance monome de P < puissance monome de Q alors
      inserer monome de P dans la liste resultat;
      passer au monome suivant de P;
    sinon
      inserer monome de Q dans la liste resultat;
      passer au monome suivant de Q;
    fin si;
  fin si;
fin tantque;

tantque il reste un monome dans P faire
  inserer le monome courant dans resultat;
  passer au monome suivant de P;
fin tantque;

tantque il reste un monome dans Q faire
  inserer le monome courant dans resultat;
  passer au monome suivant de Q;
fin tantque;
```

Listing 2 – Algorithme général

```
procedure additionner (entree p <Liste[Monome]>,
             entree q <Liste[Monome]>,
             sortie somme <Liste[Monome]>)
  declenche listePleine

glossaire
  i <Entier>; --indice de parcours de p
  j <Entier>; --indice de parcours de q
  k <Entier>; --indice de parcours de somme
  e1 <Monome>;
  e2 <Monome>;
  e3 <Monome>;
debut
  creerListeVide(somme);
```

```
15    k <- 1;
16    i <- 1;
17    j <- 1;
18    tantque i <= longueur(p) et j <= longueur(q) faire
19      e1 <- ieme(p, i);
20      e2 <- ieme(q, j);
21      si exposant(e1) = exposant(e2) alors
22        e3 <- somme(e1, e2);
23        si coefficient(e3) /= 0 alors
24          inserer(somme, k, e3);
25          k <- k + 1;
26        fin si;
27        i <- i + 1;
28        j <- j + 1;
29      sinon
30        si exposant(e1) < exposant(e2) alors
31          inserer(somme, k, e1);
32          k <- k + 1;
33          i <- i + 1;
34        sinon
35          inserer(somme, k, e2);
36          k <- k + 1;
37          j <- j + 1;
38        fin si;
39      fin si;
40    fin tantque;
41
42    tantque i <= longueur(p) faire
43      inserer(somme, k,ieme(p,i));
44      i <- i + 1;
45    fin tantque;
46
47    tantque j <= longueur(q) faire
48      inserer(somme, k, ieme(q,j));
49      j <- j + 1;
50    fin tantque;
51  fin
52
53  -- logiscop
```

Listing 3 – Programme