4TIN302U: Algo des structures élémentaires: feuille 1

Test No 1, 3/11/2016, 14h, Durée 30mn

1 Polynômes

On considère des polynômes à coefficients entiers :

$$P(x) = a_0 * x^0 + a_1 * x^1 + \ldots + a_{N-1} * x^{N-1}$$

On définit un type abstrait pour manipuler ces polynômes par l'ensemble de primitives suivant :

Primitives du TDA polynôme

```
1 polynome polynome_creer();
2 void polynome_supprimer(polynome p);
3 void polynome_afficher(polynome p);
4 void polynome_modifier_monome(polynome p, int n, int a);
5 polynome polynome_additionner(polynome p1, polynome p2);
6 polynome polynome_soustraire(polynome p1, polynome p2);
```

Ci-dessous quelques exemples de manipulations :

$$P(x) = 1 * x^0 + 2 * x^1 - 3 * x^3$$

$$Q(x) = 2 * x^0 - 1 * x^1$$

$$R(x) = P(x) + Q(x) = 3 * x^0 + 1 * x^1 - 3 * x^3$$

$$S(x) = P(x) - Q(x) = -1 * x^0 + 3 * x^1 + 3 * x^3$$

2 Je suis concepteur

Exercice 1. Soit une première implémentation des polynômes $P(x)$ par $p[N]$, des tableaux d'entiers, en posant $p[j] = a[j]$.

Complétez le code ci-dessous :

Primitives du TDA polynôme

```
1  #define N 10
2  typedef int* polynome;
3  polynome
4  polynome_creer(){
5    polynome p= calloc(N, sizeof(int) * N);
6    assert(p != NULL);
7    return p;
8  }
9
10 void
11 polynome_supprimer(polynome p){
12   if(p != NULL)
13     free(p);
14 }
15
16
17 void
18 polynome_afficher(polynome p){
19   printf(" P(x)= ");
20   int i=0;
```

```
  while(p[i] == 0 && i<N){
    printf("\t");
    i++;
  }
  if(i == N)
    {
      printf(" 0 ");
    }
  else
    {
      if(p[i] < 0)
        printf("%d * x^{%d}", p[i], i);
      else
        printf(" %d * x^{%d}", p[i], i);
      for(int j=i+1; j<N; j++){
        if(p[j] == 0)
          printf("\t     ");
        else if (p[j] < 0)
          printf("   %d * x^{%d}", p[j], j);
        else
          printf(" +  %d * x^{%d}", p[j], j);
      }
    }
  printf("\n");
}


void
polynome_modifier_monome(polynome p, int n, int a){
  assert(n < N);
  p[n]= a;
}

polynome
polynome_additionner(polynome p1, polynome p2){
 \\ A FAIRE
}

polynome
polynome_soustraire(polynome p1, polynome p2){
 \\ A FAIRE
}
```

1. Quel est le résultat affiché par la fonction `main` donnée :

Exemple test

```
int
main(void){
  polynome p= polynome_creer();
  polynome_ajouter_monome(p, 0, 1);
  polynome_ajouter_monome(p, 1, 1);
  polynome_ajouter_monome(p, 2, -3);
  polynome_afficher(p);
  polynome q= polynome_creer();
  polynome_ajouter_monome(q, 0, 2);
  polynome_ajouter_monome(q, 1, -1);
  polynome_afficher(q);
  polynome r= polynome_additionner(p, q);
  polynome_afficher(r);
  polynome s= polynome_soustraire(p, q);
  polynome_afficher(s);
```

```
  return EXIT_SUCCESS;
}
```

3 Je suis utilisateur

Exercice 2. Soit le type SList, liste simplement chaînée, défini comme il suit :

SList

```
typedef int data_type;
typedef struct SList *SList;
extern SList asde_slist_alloc(void);
extern  void asde_slist_free_link(SList link_);
extern SList asde_slist_prepend(SList L, data_type data);
extern SList asde_slist_delete_first(SList L);
extern SList asde_slist_insert_after(SList L, SList p, data_type data);
extern SList asde_slist_delete_after(SList L, SList p);
extern  SList  asde_slist_next(SList L);
extern  data_type asde_slist_data(SList L);
```

1. Soit L, $L = \{1, 1, 0, 0, 1, 0, 1\}$, une liste simplement chaînée d'entiers, représentée par `SList`, et la fonction `mystere` suivante :

mystere

```
SList
mystere(SList L, data_type data){
  if(L == NULL)
    return NULL;
  if(asde_slist_data(L) == data)
    return L;
  return mystere(asde_slist_next(L), data);
}
```

Quel est le contenu de la liste `l` résultat de l'appel à la fonction mystère suivant :

mystere

```
l= mystere(L, 0)
```

2. On souhaite implémenter les polynômes d'entiers par `SList`, le type de listes simplement chaînées.
   a. Donnez la défintion du type `polynome` .
   b. Proposez l'implémentation de deux primitives au choix parmi celles données dans la première section.