Université de Bordeaux — Licence Maths/Informatique : $2^{ème}$ année
Algorithmique des structures de données élémentaires — C. Blanc, M. Bousabbah
DS1 -25 octobre 2016- 9h30 — Durée : 30mn

Nombres binaires : Un système binaire est un système de numérotation utilisant la base 2. On nomme couramment un bit les chiffres de la numérotation binaire positionnelle. Ceux-ci ne peuvent prendre que deux valeurs, notées par convention 0 et 1.
L'écriture binaire repose sur le fait que tout nombre peut s'écrire sous la forme d'une somme de puissance de 2.
Par exemple, l'écriture binaire de **23** en décimal est **1 0 1 1 1**.

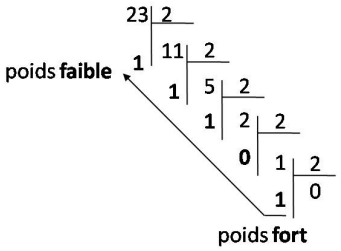

FIGURE 1 – Conversion en nombre binaire

On propose de représenter un nombre binaire par une liste simplement chainée ayant en première position le bit de poids faible. Le nombre 0 sera réprésenté par une liste d'un élément de valeur 0. On utilise la structure de données suivante :

```
struct SList_Bin{
  int bit;
  struct SList_Bin* next;
};
typedef struct SList_Bin* SList_Bin;
```

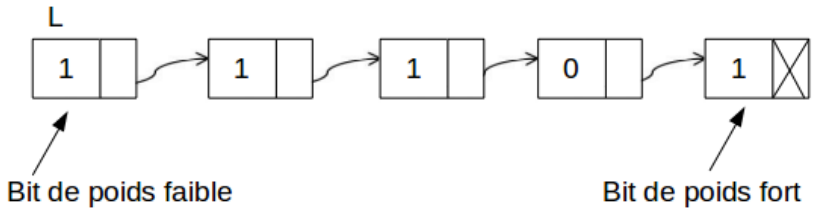

FIGURE 2 – Représentation d'un nombre binaire par une liste simplement chainée

On dispose des primitives du type abstrait listes simplement chainées vues en cours et dont les prototypes sont rappelés en annexe.

Exercice 1 *Ecrivez les fonctions suivantes :*

1. *`SList_Bin multiplierPar2(SList_Bin L)`*
   qui multiplie par 2 le nombre binaire représenté par L

   ```
   asde_slist_prepend(L,0)
   ```

2. *`SList_Bin diviserPar2(SList_Bin L)`*
   qui divise par 2 le nombre binaire représenté par L

   ```
   asde_slist_delete_first(L)
   ```

3. *`bool estPair(SList_Bin L)`*
   qui vaut `True` si L représente un nombre pair et `False` sinon.

   ```
   return (asde_slist_data(L)==0)
   ```

4. *Pour chaque fonction précisez sa complexité.*
   `Chaque fonction est O(1) car l'action se situe en tête de liste.`

Exercice 2 *On souhaite compléter l'implémentation en ajoutant aux primitives connues les deux suivantes :*

— *`void setData(SList_Bin L, bit x);`*
qui remplace par x la valeur du bit pointé par L.
— *`void setNext(SList_Bin L, SList p);`*
qui remplace la valeur du suivant de L par p.

1. *Ecrivez le code des primitves `setData` et `setNext`.*

   ```
   void setData(SList_Bin L, bit x){
     L->bit=p;
   }

   void setNext(SList_Bin L,SList p){
     L->next=x;
   }
   ```

2. *Donnez la complexité des fonctions. La complexité est en O(1) car on agit sur le pointeur L sans parcourir la liste.*

Incrémentation On dispose d'une fonction `Inc_Bin` qui permet d'incrémenter un nombre binaire, c'est à dire d'ajouter 1 au nombre donné en paramètre. Par exemple :
L'incrémentation de **1 0 1 1 1** est égale à **1 1 0 0 0** ;
L'incrémentation de **1 1 0 0 0** est égale à **1 1 0 0 1** ;
L'incrémentation de **11** est égale à **100**.

Attention avec notre implémentation les bits de poids faibles d'un nombre binaire sont en tête de liste. Par exemple l'incrémentation de **10111** (FIGURE 2) donnera **11000** qui sera donc réprésenté par la liste suivante :

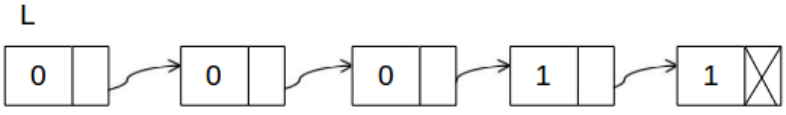

FIGURE 3 – Représentation du nombre binaire après incrémentation

On vous propose une version itérative de la fonction Inc_Bin ci-dessous. Pour faciliter la lecture de ce code nous avons supprimé le préfixe asde_slist des primitives.

Inc_Bin

```
SList_Bin Inc_Bin(SList_Bin L)
{
  if(L==NULL) return NULL;
  else
 {  if (getData(L)==0)
      setData(L,1);
    else {
      setData(L,0);
      SList_Bin p=getNext(L);
      while(getNext(p)!=NULL && getData(p)==1 )
                { setData(p,0);
                  p=getNext(p);
                }
      if (getData(p)==0)
         setData(p,1);
      else //Ici on a : p->next==NULL and p->bit==1
                {
        setData(p,0)
        Insert_After(L,p,1)
                }
         }
  return L;
 }
}
```

Exercice 3 *En utilisant les primitives du type abstrait listes simplement chainées et éventuellement les nouvelles primitives :*

1. *Complétez le code itératif de cette fonction (zone TODO).*
2. *Ecrivez une version récursive de la même fonction.*

```
SList_Bin Inc_Bin(SList_Bin L){
  if(L==NULL)
  return NULL;
if getData(L)==0 {
  setData(L,1);
return L;
}
  setData(L,0);
if (getNext(L)==NULL){
  L=insert_After(L,L,1);
return L;
}
SList_Bin p =return(Inc_Bin(getNext(L));
setNext(L,p);
return L;
}
```

3. *Pour chaque version donnez la complexité de vos fonctions.* *Dans les 2 cas on doit parcourir la liste dans le pire de cas donc O(n) si n est le nombre d'éléments de L.*

-

Annexe : Ci-dessous le fichier `asde_slist.h` décrit l'interface du type abstrait asde_slist

asde_slist.h

```
typedef int data_type;
typedef struct SList *SList;
// allocates space for one SList element
 SList asde_slist_alloc(void);

// frees space for one SList element
  void asde_slist_free_link(SList link_);

//adds a new element on the start of the list
 SList asde_slist_prepend(SList L, data_type d);

// delete an element on the start of the list
 SList asde_slist_delete_first(SList L);

// insert un new element after SList p
SList asde_slist_insert_after(SList L, SList p, data_type d);

// delete an element after SList p
 SList asde_slist_delete_after(SList L, SList p);

// gets next element in a SList
  SList  asde_slist_getNnext(SList L);

// gets data in a SList
  data_type asde_slist_getData(SList L);
```