

**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**.

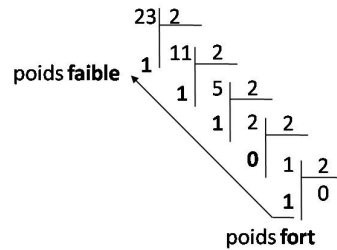


FIGURE 1 – Conversion en nombre binaire

On propose de représenter un nombre binaire par une liste simplement chaînée ayant en première position le bit de poids faible. Le nombre 0 sera représenté 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;
    
```

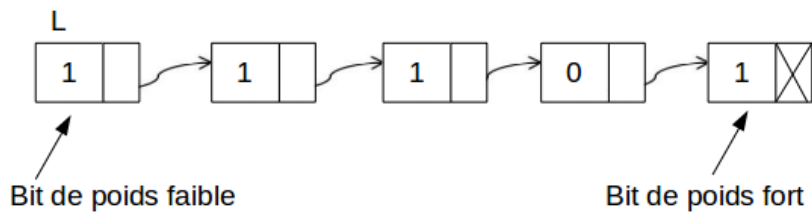


FIGURE 2 – Représentation d'un nombre binaire par une liste simplement chaînée

On dispose des primitives du type abstrait listes simplement chaînées 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*
2. *SList\_Bin diviserPar2(SList\_Bin L)*  
*qui divise par 2 le nombre binaire représenté par L*
3. *bool estPair(SList\_Bin L)*  
*qui vaut True si L représente un nombre pair et False sinon.*
4. *Pour chaque fonction précisez sa complexité.*

**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 primitives setData et setNext.*
  2. *Donnez la complexité des fonctions.*

**Incréméntation** 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'incréméntation de **1 0 1 1 1** est égale à **1 1 0 0 0**;

L'incréméntation de **1 1 0 0 0** est égale à **1 1 0 0 1**;

L'incréméntation 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'incréméntation de **10111** (FIGURE 2) donnera **11000** qui sera donc représenté par la liste suivante :

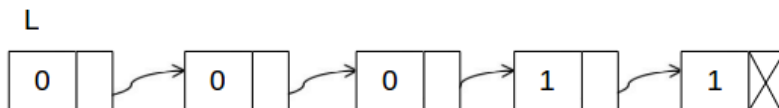


FIGURE 3 – Représentation du nombre binaire après incréméntation

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
        {
          //TODO
        }
    }
  }
  return L;
}
```

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

1. *Complétez le code itératif de cette fonction (zone TODO).*
2. *Ecrivez une version récursive de la même fonction.*
3. *Pour chaque version donnez la complexité de vos fonctions.*



**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);
```