École de l'air 2005

Avertissements :

- Il sera tenu compte dans le barême de la clarté et de qualité de la rédaction.
- Avant de donner un programme, on commence par décrire brièvement l'algorithme utilisé.

Notations :

- On considère l'algèbre de Boole $\mathcal{B} = (\{0,1\}, +, .)$ munie des opérations usuelles. On note pour $x \in \mathcal{B}$, $\overline{x} = 1 - x$.
- La suite de Fibonacci est une suite d'entiers naturels définie par $F_0 = 1$, $F_1 = 2$ et $F_{n+2} = F_{n+1} + F_n$ pour tout $n \in \mathbb{N}$.

1 Partie I : Logique

Soit $n \in \mathbb{N}^*$. Un *additionneur n-bits* est un circuit qui prend en entrée deux opérandes $a = (a_0, ..., a_{n-1})$ et $b = (b_0, ..., b_{n-1})$ ainsi que la *retenue entrante* r et qui rend en sortie la somme $s = (s_0, ..., s_n)$ donnée par

$$\sum_{k=0}^{n} 2^k s_k = \sum_{k=0}^{n-1} 2^k a_k + \sum_{k=0}^{n-1} 2^k b_k + r.$$

On dit alors que s_n est la *retenue sortante*. Cette formule correspond à l'addition de deux nombres binaires et les éléments a_k, b_k, s_k et r sont des entiers de l'ensemble $\{0,1\}$.

1) On commence par le cas où $n = 1$. Donner, sous forme d'un tableau, s_0 et s_1 en fonction de a_0, b_0 et r.
2) En déduire les expressions de s_0 et s_1 en fonction de a_0, b_0 et r dans l'algèbre de Boole $\mathcal{B}$.
3) Comment construire un additionneur n-bits à partir de n additionneurs 1-bit ?

2 Partie II : Nombre de Fibonacci

On considère le programme suivant qui calcule les nombres de Fibonacci :

• En Caml :

```
let rec fibo = function
    |0 -> 1
    |1 -> 2
    |n -> fibo (n-1) + fibo(n-2);;

fibo : int -> int = <fun>
```

• En Pascal :

```
function fibo ( n : integer ) : integer;
begin
    if n = 0 then fibo := 1
    else if n = 1 then
           fibo := 2
         else
           fibo := fibo ( n - 1 ) + fibo ( n - 2 );
end;
```

1) Déterminer la complexité de `fibo` : on prendra en compte uniquement le nombre total d'additions nécessitées par les appels récursifs.
2) Représenter les appels récursifs de `fibo(5)` sous forme d'un arbre binaire.
3) Comment améliorer l'algorithme pour obtenir une complexité linéaire ? Justifier votre réponse.

3 Partie III : Numération dans la base de Fibonacci

Pour k et m deux entiers, on note $k >> m$ quand $k \geqslant m+2$.
Soit $n \in \mathbb{N}^*$. Une décomposition de n dans *la base de* Fibonacci est la donnée de $r \in \mathbb{N}$ et d'un $(r+1)$-uplet $(i_r, ..., i_0)$ d'entiers naturels tels que

$$n = \sum_{k=0}^{r} F_{i_k}$$

avec $i_r >> i_{r-1} >> ... >> i_0$.

1) Soit $n \in \mathbb{N}^*$. Montrer qu'il existe une unique décomposition de n dans la base de Fibonacci.
2) Déterminer cette décomposition pour 67.

Les trois questions suivantes s'adressent **uniquement** aux candidats qui programment en Caml :

3) Écrire une fonction
`listefibo : int -> int * int list`
qui prend en argument un entier $n \in \mathbb{N}^*$ et qui renvoie un couple formé de l'indice p et de la liste $(F_p, F_{p-1}, ..., F_0)$ avec p tel que $F_p \leqslant n < F_{p+1}$.
4) Écrire une fonction
`decompose : int -> int list`
qui prend en argument un entier $n \in \mathbb{N}^*$ et qui renvoie l'unique décomposition de n dans la base de Fibonacci. On pourra s'inspirer de la division euclidienne.

Soit $n \in \mathbb{N}^*$. À la décomposition de n dans la base de Fibonacci sous la forme $(F_{i_r}, F_{i_{r-1}}, ..., F_{i_0})$, avec $i_r = p$, on associe un mot $u(n)$ sur l'alphabet $A = \{0, 1\}$ tel que $u(n) = u_p u_{p-1} ... u_1 u_0$ avec $u_i = 1$ si et seulement si F_i apparaît dans la décomposition de n dans la base de Fibonacci. Par exemple on a $63 = F_8 + F_4$ donc $u(63) = 100010000$.

5) Écrire une fonction
`motdanslabase : int -> int list`
qui prend en argument un entier $n \in \mathbb{N}^*$ et qui renvoie la liste correspondante au mot $u(n)$.

Les trois questions suivantes s'adressent **uniquement** aux candidats qui programment en Pascal :

On définit un type `liste` :

```
type
    liste = ^cellule;
    cellule = record
        contenu : integer; suivant : liste;
    end;
```

En Pascal, la liste vide est `nil` et l'on pourra utiliser la fonction suivante pour construire les listes :

```
function cons(p:integer ; suivant : liste) :liste ;
var r : liste ;
begin
    new(r);
    r^.contenu := p;
    r^.suivant := suivant;
```

```
    cons := r;
end ;
```

3) Écrire une fonction
`function listefibo ( n : integer ) : liste ;`
qui prend en argument un entier $n \in \mathbb{N}^*$ et qui renvoie la liste $(p, F_p, F_{p-1}, ..., F_0)$ où p est l'entier naturel tel que $F_p \leqslant n < F_{p+1}$.

4) En déduire une fonction
`function decompose ( n : integer ) : liste ;`
qui prend en argument un entier $n \in \mathbb{N}^*$ et qui renvoie l'unique décomposition de n dans la base de Fibonacci. On pourra s'inspirer de la division euclidienne.

Soit $n \in \mathbb{N}^*$. À la décomposition de n dans la base de Fibonacci sous la forme $(F_{i_r}, F_{i_{r-1}}, ..., F_{i_0})$, avec $i_r = p$, on associe un mot $u(n)$ sur l'alphabet $A = \{0, 1\}$ tel que $u(n) = u_p u_{p-1} ... u_1 u_0$ avec $u_i = 1$ si et seulement si F_i apparaît dans la décomposition de n dans la base de Fibonacci. Par exemple on a $63 = F_8 + F_4$ donc $u(63) = 100010000$.

5) Écrire une fonction
`function motdanslabase ( n: integer ) : liste;`
qui prend en argument un entier $n \in \mathbb{N}^*$ et qui renvoie la liste correspondante au mot $u(n)$.

6) On définit le langage $\mathcal{L} = \{u(n)\ ,\ n \in \mathbb{N}^*\}$. Montrer que $\mathcal{L}$ est exactement l'ensemble des mots sur A commençant par 1 et ne comportant pas de facteur 11 c'est à dire deux chiffres 1 consécutifs.

7) $\mathcal{L}$ est-il un langage rationnel ? Si oui, donner un automate fini qui reconnaît ce langage ainsi qu'une expression rationnelle qui dénote ce langage.

4 Partie IV : Somme de deux nombres dans la base de Fibonacci

Le but de cette partie est d'écrire un programme donnant l'unique décomposition de $n + n'$ connaissant la décomposition de n et de n'.

1) Soit $n = F_{i_s} + F_{i_{s-1}} + ... + F_{i_0}$ une somme de nombres de Fibonacci avec $i_s \geqslant i_{s-1} \geqslant ... \geqslant i_1 \geqslant i_0$. Décrire un algorithme qui renvoie la liste correspondant à l'unique décomposition de n.

La question suivante s'adresse **uniquement** aux candidats qui programment en Caml :

2) Écrire une fonction
`somme : int list -> int list -> int list`
qui prend en arguments deux listes correspondant à l'unique décomposition de deux entiers et qui renvoie l'unique décomposition de la somme. On pourra utiliser l'algorithme de la question précédente.

La question suivante s'adresse **uniquement** aux candidats qui programment en Pascal :

2) Écrire une fonction
`somme ( nombre1 : liste ; nombre2 : liste ) : liste`
qui prend en arguments deux listes correspondant à l'unique décomposition de deux entiers et qui renvoie l'unique décomposition de la somme. On pourra utiliser l'algorithme de la question précédente.