ÉCOLE POLYTECHNIQUE **FILIÈRE MP**

CONCOURS D'ADMISSION 2000

COMPOSITION D'INFORMATIQUE

(Durée : 4 heures)

L'utilisation des calculatrices n'est pas autorisée

⋆ ⋆ ⋆

Avertissement. On attachera une grande importance à la clarté, à la précision et à la concision de la rédaction.

Introduction. Nous considérons un système commandé par un tableau de bord comportant N interrupteurs, chacun pouvant être baissé ou levé. On désire tester ce système (pour le valider ou pour effectuer une opération de maintenance) en essayant mécaniquement chacune des 2^N configurations possibles pour l'ensemble des interrupteurs. Le coût de cette opération, qu'elle soit réalisée par un opérateur humain ou par un robot, sera le nombre total de mouvements d'interrupteurs nécessaires. Nous supposons que chaque fois qu'un interrupteur est commuté le système effectue un diagnostic automatiquement et instantanément. Les interrupteurs sont indexés de 0 à $N-1$.

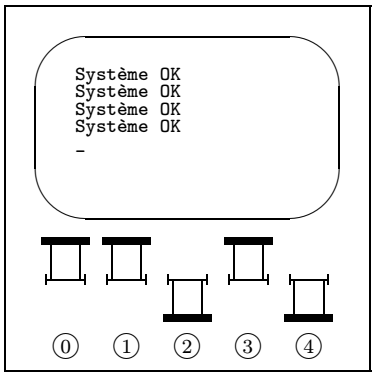

FIGURE 1. Pupitre de commande ; les interrupteurs 2 et 4 sont baissés.

Notations. Nous appelons *partie* un sous-ensemble fini de l'ensemble **N** des entiers naturels. Un élément d'une partie est appelé *indice*. La *différence symétrique* de deux parties P et Q est définie par :

$$P \triangle Q = (P \setminus Q) \cup (Q \setminus P) = (P \cup Q) \setminus (P \cap Q)$$

On vérifie facilement que la différence symétrique est commutative et associative, ce que l'on ne demande pas de démontrer. Pour tout entier n positif ou nul, nous notons $I_n = \{0, .., n-1\}$ l'ensemble des entiers inférieurs strictement à n.

Une partie sera représentée par une liste chaînée d'indices distincts apparaissant dans l'ordre croissant des entiers. On utilisera les types suivants :

Caml

```
type indice == int ;;
type partie == indice list ;;
```

Pascal

```
type indice = integer ;
type partie = ^noeud ;
noeud = record
  valeur : indice ; suivant : partie ;
end ;
```

Les candidats composant en Pascal, pourront utiliser le constructeur suivant :

```
function cree_partie (v : indice ; s : partie) : partie ;
var p : partie ;
begin new (p) ; p^.valeur := v ; p^.suivant := s ; cree_partie := p end ;
```

Première partie. Parties d'un ensemble

Question 1. Écrire la fonction `card` qui retourne le nombre d'éléments d'une partie.

Caml	Pascal
`value card : partie -> int`	`function card (p : partie) : integer ;`

Les candidats composant en Caml devront résoudre cette question sans faire appel à la fonction de bibliothèque `list_length`.

Question 2. Écrire la fonction `delta` qui réalise la différence symétrique de 2 parties. Le nombre d'opérations ne devra pas excéder $O(m+n)$, où m et n sont les cardinaux des arguments.

Caml	Pascal
`value delta : partie -> partie -> partie`	`function delta (p,q : partie) : partie ;`

Nous rappelons que dans toute liste chaînée de type `partie` les indices sont distincts et doivent apparaître dans l'ordre croissant des entiers.

Question 3. Application au problème des interrupteurs : à chacune des configurations possibles, nous associons la partie formée des indices des interrupteurs baissés.

Écrire un programme qui imprime la liste des indices d'interrupteurs à commuter pour passer d'une configuration à une autre.

Caml	Pascal
`value test : partie -> partie -> unit`	`procedure test (p,q : partie) ;`

Pour imprimer un entier `i` suivi d'un espace ou pour imprimer un saut de ligne, on pourra utiliser respectivement les instructions suivantes :

Caml	Pascal
`printf "%d " i ;`	`write (i,' ') ;`
`print_newline () ;`	`writeln ;`

Deuxième partie. Énumération des parties par incrément

À toute partie P, nous associons l'entier $e(P) = \sum_{i \in P} 2^i$ (avec la convention $e(\emptyset) = 0$). Nous définissons le *successeur* de P comme l'unique partie dont l'entier associé est $e(P)+1$.

Question 4. Écrire la fonction `succ` qui retourne le successeur d'une partie. Le nombre d'opérations ne devra pas excéder $O(l)$ où l est le plus petit indice absent dans la partie donnée en argument.

Caml	Pascal
`value succ : partie -> partie`	`fonction succ (p : partie) : partie ;`

Question 5. En application de ce mode d'énumération des parties, nous voulons réaliser le test de toutes les configurations d'interrupteurs. Au début et à la fin du test tous les interrupteurs seront levés.

a) Écrire un programme qui imprime la liste des indices des interrupteurs à commuter pour réaliser la totalité du test pour N interrupteurs et qui examine les configurations dans l'ordre défini par le successeur. L'argument de cette fonction sera l'entier N.

Caml	Pascal
`value test_incr : int -> unit`	`procedure test_incr (n : integer) ;`

b) Exprimer, en fonction de N, le nombre total d'interrupteurs à commuter pour réaliser le test de cette manière.

Troisième partie. Énumération des parties par un code de Gray

Nous notons $\langle u_0, \ldots, u_{l-1} \rangle$ une suite finie de l entiers. La concaténation de deux suites finies de longueur l et l' respectivement est une suite finie de longueur $l + l'$ définie par

$$\langle u_0, \ldots, u_{l-1} \rangle \odot \langle u'_0, \ldots, u'_{l'-1} \rangle = \langle u_0, \ldots, u_{l-1}, u'_0, \ldots, u'_{l'-1} \rangle.$$

La suite vide, notée $\langle\,\rangle$, est la suite de longueur 0. Une suite finie U est *préfixe* d'une autre suite finie V s'il existe une suite finie W telle que $V = U \odot W$ (autrement dit U est le début de V). Pour tout entier positif ou nul n, nous considérons la suite finie $T(n)$ de longueur $2^n - 1$ définie par

$$T(0) = \langle\,\rangle \text{ et } T(n+1) = T(n) \odot \langle n \rangle \odot T(n).$$

Nous avons par exemple $T(1) = \langle 0 \rangle$, $T(2) = \langle 0, 1, 0 \rangle$ et $T(3) = \langle 0, 1, 0, 2, 0, 1, 0 \rangle$.

Pour tout entier i positif ou nul nous notons t_i le $(i+1)$-ème élément de $T(n)$ s'il existe. Puisque $T(n)$ est préfixe de $T(n+1)$, la suite $(t_i)_{i \geqslant 0}$ est définie sans ambiguïté. Enfin, nous posons $S_0 = \emptyset$ et pour tout entier i positif ou nul nous définissons l'ensemble $S_{i+1} = S_i \triangle \{t_i\}$.

Question 6.

a) Donner la valeur de $T(4)$.

b) Donner la valeur de S_i pour tout i inférieur ou égal à 15.

Question 7. Nous voulons montrer que les S_i peuvent être utilisés pour énumérer les parties de I_n, et ainsi résoudre notre problème d'interrupteurs.

a) Donner la valeur de S_{2^n-1} pour tout $n \geqslant 0$.

b) Montrer que pour tout $n > 0$ et tout $i < 2^n$, on a $S_{2^n+i} = S_i \triangle \{n-1, n\}$.

c) En déduire que pour tout $n \geqslant 0$ l'ensemble $\mathcal{P}_n = \{S_0, S_1, \ldots, S_{2^n-1}\}$ est l'ensemble des parties de I_n.

Question 8. Application au problème des interrupteurs. Comme dans la Deuxième partie, nous imposons que les interrupteurs soient levés au début et à la fin du test.

a) Écrire un programme s'inspirant des résultats de cette partie, qui imprime une liste d'indices d'interrupteurs à commuter pour réaliser la totalité du test. L'argument de cette fonction sera le nombre d'interrupteurs N.

Caml	Pascal
`value test_gray : int -> unit`	`procedure test_gray (n : integer) ;`

b) Quel est le coût du test avec cette méthode (c'est-à-dire le nombre total d'interrupteurs à commuter) ? Peut-on réaliser le test à un coût moindre ?

Question 9. Successeur de Gray. Pour tout $i > 0$, on note $\min(S_i)$ le plus petit élément de S_i.

a) Donner une expression de t_i en fonction de S_i pour i impair.

b) Écrire la fonction `gray` qui prend en argument une partie et retourne celle qui la suit immédiatement dans l'ordre défini par la suite $(S_i)_{i\geqslant 0}$.

Caml	Pascal
`value gray : partie -> partie`	`fonction gray (p : partie) : partie ;`

Quatrième partie. Système défaillant

Chaque interrupteur baissé active une composante du système, et un mauvais fonctionnement de l'alimentation électrique provoque une défaillance dès que plus de K interrupteurs sur les N sont baissés.

Question 10. Écrire un programme qui imprime une liste d'interrupteurs à commuter, de taille minimale, permettant de passer d'une configuration non défaillante à une autre sans provoquer de défaillance. Les arguments de ce programme seront la partie de départ et la partie cible.

Caml	Pascal
`value test_sur : partie -> partie -> unit`	`procedure test_sur (p,q : partie) ;`

Question 11. L'inverse d'une suite finie T est obtenue en prenant ses éléments dans l'ordre inverse, nous la notons $\widetilde{T}$. Soit $T(n,k)$ la suite définie pour tout $k \geqslant 1$ et pour tout $n \geqslant 1$ par

$$\begin{aligned} T(1,k) &= \langle 0 \rangle \\ T(n+1,1) &= T(n,1) \odot \langle n-1, n \rangle \\ T(n+1,k+1) &= T(n,k+1) \odot \langle n \rangle \odot \widetilde{T}(n,k) \end{aligned}$$

Soit k un entier strictement positif. Pour tout entier i positif ou nul nous notons $t_{k,i}$ le $(i+1)$-ème élément de $T(n,k)$ s'il existe. Puisque $T(n,k)$ est préfixe de $T(n+1,k)$, la suite $(t_{k,i})_{i\geqslant 0}$ est définie sans ambiguïté. Enfin, nous posons $S_{k,0} = \emptyset$ et pour tout entier i positif ou nul nous définissons l'ensemble $S_{k,i+1} = S_{k,i} \vartriangle \{t_{k,i}\}$.

a) Exprimer la longueur $l_{n,k}$ de $T(n,k)$ en fonction de n, de k et du nombre $s_{n,k}$ de parties de I_n de cardinal inférieur ou égal à k.

b) Montrer que pour tout $k \geqslant 1$ et tout $n \geqslant 1$ l'ensemble $\mathcal{P}_{n,k} = \{S_{k,0}, S_{k,1}, \ldots, S_{k,l_{n,k}}\}$ est l'ensemble des parties de I_n de cardinal inférieur ou égal à k.

Question 12. Écrire un programme qui affiche une liste de $l_{N,K}+1$ interrupteurs à commuter permettant de vérifier toutes les configurations non défaillantes sans provoquer de défaillance, en commençant et en finissant avec des interrupteurs tous levés. Les entiers N et K sont donnés en arguments.

Caml	Pascal
`value test_panne : int -> int -> unit`	`procedure test_panne (n,k : integer) ;`

Question 13. Montrer que le coût d'un test commençant et en finissant avec des interrupteurs tous levés et ne provoquant pas de défaillance ne peut pas être inférieur à $l_{N,K}+1$.

* *
*