INFORMATIQUE

Les deux parties sont indépendantes et peuvent être traitées dans un ordre quelconque.

Partie I - Un algorithme de compression de données

Dans cette partie, on appelle *document* une suite de symboles terminée par un symbole particulier que l'on note `SymboleFinal`. On s'intéresse à la représentation d'un document sous la forme la plus compacte possible.

I.A - Codage de longueurs de séquences

Une technique de réduction du nombre de symboles utilisés pour représenter un document consiste à remplacer chaque séquence de symboles identiques par la longueur de la séquence suivie du symbole. Il faut toutefois pouvoir distinguer les longueurs de séquences des symboles originellement présents dans le document. On utilise pour cela des délimiteurs spécifiques, ne figurant pas parmi les symboles présents dans le document.

L'ensemble des symboles du document est un ensemble de caractères qui contient les lettres minuscules et majuscules, les chiffres décimaux, les caractères de ponctuation et divers caractères spéciaux : parenthèses, accolades, guillemets **mais ne contient pas les crochets**.

On choisit alors les crochets [et] comme délimiteurs des longueurs de séquences et on peut ainsi coder, par exemple, le document suivant :

```
abcdeffffghiiiiiijkllllllllll
```

sous la forme :

```
abcde[5]fgh[6]ijk[10]l
```

En fait, à chaque symbole est associé un entier que l'on appelle son code, qui le représente dans le document codé. Ainsi le document ci-dessus sera-t-il codé en réalité sous la forme :

```
a'b'c'd'e'[5]f'g'h'[6]i'j'k'[10]l'
```

où `a'` est le code de `a`, `b'` est le code de `b`, etc.

On dispose des fonctions suivantes :

`symbole_suivant` rend l'entier correspondant au symbole suivant du document, ou le code `SymboleFinal` lorsque la fin du document est atteinte ;

Filière MP

sortir_char prend en argument un caractère et le place en sortie du codeur (cette sortie peut être l'écran ou un fichier) ;

sortir_code prend en argument un entier et place en sortie du codeur le symbole (caractère) dont cet entier est le code ;

sortir_int prend en argument un entier et place en sortie du codeur la suite de symboles qui représente cet entier en base 10 ;

symbole prend en argument un entier et rend le symbole dont cet entier est le code ;

code prend en argument un symbole et rend son code.

Note : on rappelle que les délimiteurs ouvrant et fermant de longueur de séquence [et] n'apparaissent pas dans le document initial à coder.

On définit une structure de données `Encodeur` permettant de représenter, sous forme d'un enregistrement, l'état du codeur de longueurs de séquences. Cet état correspond à la séquence de symboles qu'il est en train de lire et à la position à laquelle il se trouve dans cette séquence.

en CAML

```
type Encodeur =
   {mutable code_symbole : int;
    mutable compte : int
   }
```

en PASCAL

```
Encodeur = record
   code_symbole : integer;
   compte : integer
end;
```

I.A.1) Écrire une fonction `initialiser_codeur` qui rend un `Encodeur` représentant l'état initial d'un codeur (hors de toute séquence de symboles).

Écrire une fonction `vider_codeur` qui reçoit en argument un `Encodeur` et place sur la sortie le résultat du codage de la séquence de symboles qui correspondent à son état. Cette fonction rend le nouvel état du codeur.

I.A.2) Écrire une fonction `coder` qui reçoit en argument un entier et un `Encodeur` et rend un `Encodeur` correspondant à l'état du codeur après traitement du symbole dont l'entier est le code.

I.A.3) Quel est le résultat du codage du document `aabbcc` avec cet algorithme ? Quel problème cela pose-t-il ? Que peut-on modifier pour remédier à ce problème ?

I.A.4) Écrire une fonction `decoder` qui, lorsqu'elle lit un document codé, affiche en sortie un document dans lequel les séquences compactées sous la forme `[compte]code_symbole` sont restaurées sous leur forme initiale.

On supposera que le document lu est correct, c'est-à-dire que toute occurrence du symbole [est suivie d'une suite de chiffres terminée par], elle-même suivie d'un entier, code d'un symbole.

On supposera de plus que les caractères 0 à 9 sont représentés par des entiers consécutifs croissants.

Partie II - Problème logique et automate

Après l'évasion de Thésée, Minos décida de modifier le labyrinthe de Dédale afin de l'utiliser pour tester les qualités de logicien des étrangers désireux d'intégrer sa cour. Pour cela, il fit modifier le tracé afin que toute personne ayant retrouvé son chemin passe dans un couloir fermé par trois portes devant impérativement être ouvertes successivement. En cas d'échec, le candidat se retrouve projeté dans une partie sans issue du labyrinthe. Chacune des trois portes est ouverte par un système de trois leviers. Ils sont tous sur une position de départ neutre et possèdent deux positions de fonctionnement 1 ou 0, correspondant respectivement aux VRAI et FAUX logiques, mais également à la numération en base 2. Ainsi, les neufs leviers (les trois de chaque porte) forment-ils un nombre écrit en base 2 lorsque le candidat les a manipulés successivement (aucun levier ne peut rester neutre). La position du levier 1 est le bit de plus haut poids, celle du levier 9 est le bit de plus faible poids. En visite dans le labyrinthe, vous vous retrouvez dans ce couloir et votre seule chance de survivre est de répondre correctement en respectant les règles propres à chaque porte et indiquées sur le fronton. Ces règles de fonctionnement de chaque porte sont systématiquement vérifiées (ce qui n'est bien sûr pas nécessairement le cas des énoncés relatifs aux positions des leviers).

Sur le fronton de la première porte est écrit : « *les trois énoncés associés aux trois leviers sont tous vrais ou tous faux* ».

Les trois énoncés sont notés respectivement E_1, E_2, E_3. Les variables propositionnelles associées aux leviers de la première porte L_1, L_2, L_3.

II.A - Représenter la règle sous la forme d'une formule du calcul des propositions dépendant de E_1, E_2, E_3.

Les énoncés suivants sont inscrits sur la porte :

- *Le levier* 2 *ne peut pas être sur «* 1 *» seul, mais les trois ne sont pas sur «* 1 *».*

- *Si le levier* 3 *est sur «* 1 *», ou si les leviers* 1 *et* 2 *sont sur «* 0 *» alors le levier* 3 *est sur «* 1 *» et ce n'est pas le seul dans ce cas.*
- *Si le levier* 1 *est sur «* 1 *» alors le levier* 3 *y est aussi, et si le levier* 1 *est sur «* 0 *» alors c'est également le cas du levier* 2 .

II.B - Exprimer E_1, E_2 et E_3 sous la forme de formules du calcul des propositions dépendant de L_1, L_2 et L_3.

II.C - En utilisant le calcul des propositions (résolution avec les formules de De Morgan, ...), simplifier les énoncés pour les écrire sous forme de conjonction (ET) de disjonctions (OU) de littéraux, un littéral étant une variable propositionnelle ou sa négation.

II.D - En déduire la ou les valeurs possibles des variables propositionnelles L_1, L_2 et L_3.

La première porte s'ouvre, puis se referme après votre passage. Elle est suivie par une deuxième porte sur le fronton de laquelle est écrit : « *Une seule des affirmations est fausse* ». Les trois énoncés sont notés respectivement E_4, E_5, E_6. Les variables propositionnelles associées aux leviers de la deuxième porte L_4, L_5, L_6.

Les énoncés suivants sont inscrits sur la porte :

- *La valeur du levier* 4 *est le produit des valeurs des leviers* 5 *et* 6 .
- *La valeur du levier* 5 *est la somme sans retenue (addition* 1 *bit) des valeurs des leviers* 4 *et* 6 .
- *La valeur du levier* 6 *est la retenue de la somme des valeurs des leviers* 4 *et* 5 .

II.E - Exprimer E_4, E_5 et E_6 sous la forme de formules du calcul des propositions dépendant de L_4, L_5, L_6.

II.F - En déduire la ou les valeurs possibles des variables propositionnelles L_4, L_5 et L_6.

Les ingénieurs crétois avaient conçu des équivalents mécaniques de nos portes logiques actuelles AND, OR et NOT.

À chaque porte est ainsi associé un circuit prenant en entrée les positions des trois leviers et donnant en sortie VRAI ou FAUX respectivement pour ouvrir ou non la porte.

II.G - Construire, en les justifiant, les circuits permettant de réaliser les opérations d'ouverture des portes, en fonction des réponses successives données.

Afin d'éviter que quelqu'un puisse réussir à sortir en testant successivement toutes les possibilités, les ingénieurs ont installé un système qui fonctionne de la manière suivante :

- dès que les trois leviers de la première porte ont été positionnés, la porte s'ouvre et les positions de ces trois leviers sont mémorisées, que ces positions soient correctes ou non ;
- la seconde porte ne s'ouvre que si les positions des six leviers sont correctes ; sinon le candidat est orienté vers une voie définitivement sans issue.

Les crétois disposent à cette fin d'un circuit mémoire à deux entrées et une sortie telle que le couple $(1, 0)$ enregistre la valeur VRAI (ou 1) et $(0, 1)$ la valeur FAUX (ou 0).

II.H - Proposer un circuit permettant de réaliser cette opération.

II.I - Utiliser ce circuit mémoire pour connecter les deux montages de la question II.G et réaliser un circuit unique ouvrant dans tous les cas la première porte, puis la seconde uniquement si toutes les réponses ont été correctes.

Après la réussite aux deux premières épreuves, le couloir débouche sur une troisième porte, sur le fronton de laquelle est écrit : « *la position des trois derniers leviers, l'un au moins n'étant pas sur* « 0 », *permet au nombre, écrit en binaire, formé par les neufs leviers d'être divisible par* 7 ».

II.J - En déduire la ou les positions possibles des leviers L_7, L_8 et L_9.

Pour vérifier si la réponse donnée est acceptable, les ingénieurs utilisent un automate fini. L'alphabet est $A = \{0, 1\}$. Les mots sont les écritures binaires des nombres, en commençant par le bit de poids fort. L'automate $\mathscr{A}$ est donc décrit par la structure $<Q, A, E, I, T>$ où :

- Q est un ensemble non vide appelé ensemble des états de $\mathscr{A}$,
- A est l'alphabet,
- E est un sous-ensemble de $Q \times A \times Q$ appelé ensemble des transitions de $\mathscr{A}$,
- I est un sous-ensemble de Q appelé ensemble des états initiaux de $\mathscr{A}$,
- T est un sous-ensemble de Q appelé ensemble des états terminaux de $\mathscr{A}$.

On représente graphiquement l'automate $\mathscr{A}$ de la façon suivante :

- un état p est figuré par un cercle marqué par p :
  - le fait que $p \in I$ est représenté par une flèche sans origine entrant dans le cercle marqué par p :
  - le fait que $p \in T$ est représenté par un double cercle autour de p ;
- une transition $(p, a, q) \in E$ est représentée par une flèche allant de l'état p vers l'état q et étiquetée par la lettre a.

II.K - Si les écritures binaires de n et n' sont $n = \overline{b_1 b_2 \ldots b_k}$ et $n' = \overline{b_1 b_2 \ldots b_k b_{k+1}}$ (b_1 étant le bit de poids fort), déterminer le reste de la division de n' par 7 connaissant le reste de la division de n par 7.

II.L - Déterminer les différentes transitions possibles entre les différents états de l'automate.

II.M - Construire, en justifiant avec soin, un automate testant la divisibilité par 7. On rappelle que les trois bits de plus faible poids ne peuvent pas être tous les trois nuls.

II.N - Modifier l'automate précédent pour tester la conformité des solutions proposées pour les leviers des trois portes.

II.O - Appliquer l'automate pour vérifier la solution proposée pour les leviers des trois portes.

••• FIN •••
