# Option Informatique en Spé MP et MP*

## Devoir surveillé du mercredi 24 mars 1999

**Résumé**

Pour changer un peu, ce sujet a été spécialement conçu pour les MP. Il est organisé en trois problèmes, comme les épreuves d'informatique des trois grands concours communs.

Le premier problème vous propose d'étudier la notion de *racine carrée* d'un langage ; une bonne occasion de mettre en œuvre vos connaissances sur les automates finis, les langages rationnels, les expressions rationnelles, le lemme de l'étoile...

Le deuxième problème vous invite à construire des circuits calculant le carré d'un nombre représenté en base deux. Ici, vous devrez exploiter le cours de première année sur les fonctions booléennes et les circuits combinatoires..

Le troisième problème présente plusieurs algorithmes de découpage d'un paragraphe de texte en lignes successives et de justification de celles-ci, en vue de leur impression ou de leur visualisation sur un écran.

*Veuillez rédiger chaque partie sur une copie séparée.*

## Table des matières

# 1 Racine carrée d'un langage

▶ Dans tout ce problème, $X$ désigne un alphabet fini. Soit $u$ un mot sur $X$ ; on note $u^2$ le carré du mot $u$ : c'est le mot $u \cdot u$ obtenu en concaténant deux copies de $u$. Soit $L$ un langage sur un alphabet $X$ ; l'ensemble des mots dont le carré appartient à $L$ est noté $\sqrt{L}$ ; ainsi $u \in \sqrt{L} \iff u^2 \in L$.

▶ Soit $\mathcal{A}$ un automate fini ; vous pourrez noter $\mathcal{L}(\mathcal{A})$ le langage (rationnel) reconnu par $\mathcal{A}$.

▶ Vous pourrez confondre une expression rationnelle et le langage rationnel qu'elle représente, sans pour autant encourir de réprimande !

***Question 1*** Déterminez $\sqrt{L}$ lorsque $X = \{a,b\}$ et $L$ est le langage décrit par l'expression rationnelle `(a+b)*b`.

***Question 2*** Déterminez $\sqrt{L}$ lorsque $X = \{a,b\}$ et $L$ est le langage décrit par l'expression rationnelle `a*ba*bb`.

***Question 3*** Exhibez deux langages $L$ et $M$ distincts, tels que $\sqrt{L}$ et $\sqrt{M}$ soient égaux, mais non vides.

▶ $\mathcal{A} = (Q, \delta, i, F)$ est un automate fini déterministe complet reconnaissant un langage rationnel $L$. Soit $q \in Q$ ; on note $\mathcal{B}_q$ l'automate obtenu à partir de $\mathcal{A}$ en remplaçant $F$ par $\{q\}$, et $\mathcal{C}_q$ l'automate obtenu à partir de $\mathcal{A}$ en remplaçant $i$ par $q$.

***Question 4*** Montrez qu'un mot $u$ appartient à $\sqrt{L}$ si et seulement si il existe un état $q$ de l'automate $\mathcal{A}$ tel que $u$ soit reconnu à la fois par $\mathcal{B}_q$ et $\mathcal{C}_q$.

***Question 5*** Montrez que, si $L$ est rationnel, alors $\sqrt{L}$ l'est aussi.

***Question 6*** Soit $n$ un naturel quelconque. Comment définiriez-vous $\sqrt[n]{L}$, et comment montreriez-vous que, si $L$ est rationnel, alors $\sqrt[n]{L}$ l'est aussi ?

▶ Dans les deux questions suivantes, $X$ désigne l'alphabet $\{0,1\}$. On note $L$ le langage formé des écritures en base 2 des naturels non nuls multiples de 5 ; par exemple, $1010 \in L$ mais $1101 \notin L$. On notera que tout mot de $L$ commence par 1.

***Question 7*** Construisez un automate fini reconnaissant $L$ ; vous justifierez rapidement le fait que votre automate réalise bien ce que l'on attend de lui.

***Question 8*** Déterminez $\sqrt{L}$.

▶ Soit $L$ un langage ; on note $L^\square = \{u^2 \mid u \in L\}$. $L^\square$ est donc l'ensemble des carrés des mots de $L$.

***Question 9*** A-t-on $\sqrt{L^\square} = L$ ?

***Question 10*** A-t-on $\left(\sqrt{L}\right)^\square = L$ ?

***Question 11*** Donnez un exemple de langage rationnel $L$ tel que $L^\square$ ne soit pas rationnel.

# 2 Circuits combinatoires pour le calcul du carré

▶ On étudie dans ce problème des circuits combinatoires (construits à partir de portes logiques ET, OU et NON) calculant le carré d'un nombre dont l'écriture en base 2 requiert $n$ chiffres au plus. Un tel circuit sera noté $\mathcal{CC}(n)$ dans la suite ; il comporte $n$ entrées notées $(a_0, \ldots, a_{n-1})$ et $p$ sorties notées $(s_0, \ldots, s_{p-1})$. Le comportement de $\mathcal{CC}(n)$ est entièrement décrit par l'équation

$$\sum_{0 \leqslant k < p} 2^k s_k = \left( \sum_{0 \leqslant i < n} 2^i a_i \right)^2$$

***Question 1*** Quel circuit *très simple* réalise $\mathcal{CC}(1)$ ?

***Question 2*** Justifiez le fait que, pour $n \geqslant 2$, $\mathcal{CC}(n)$ doit avoir exactement $2n$ sorties.

▶ On s'intéresse maintenant à la construction de $\mathcal{CC}(2)$. Pour $k \in [\![0, 3]\!]$, on note $f_k$ la fonction logique telle que $s_k = f_k(a_0, a_1)$.

***Question 3*** Dressez la table donnant les valeurs des sorties de $\mathcal{CC}(2)$ en fonction des valeurs de ses entrées.

***Question 4*** Vous constatez que $f_0(a_0, a_1) = a_0$. Donnez un raisonnement *très simple* montrant que, pour tout $n$, on a $f_0(a_0, a_1, \ldots, a_{n-1}) = a_0$.

***Question 5*** Vous constatez que $f_1(a_0, a_1) = 0$. Donnez un raisonnement *très simple* montrant que, pour tout $n \geqslant 2$, on a $f_1(a_0, a_1, \ldots, a_{n-1}) = 0$.

***Question 6*** Donnez des expressions *très simples* pour chacune des fonctions $f_2$ et $f_3$, dans le cas $n = 2$.

▶ On dispose de *portes logiques élémentaires* : ce sont la porte NON (à une seule entrée) et les portes OU et ET (à deux entrées).

***Question 7*** Réalisez $\mathcal{CC}(2)$ au moyen de portes logiques élémentaires ; les entrées de chaque porte seront placées à gauche, les sorties à droite. Combien de portes logiques utilisez-vous en tout ?

***Question 8*** Réalisez de la même façon $\mathcal{CC}(3)$ au moyen de portes logiques élémentaires ; vous commencerez par dresser la table donnant les valeurs des sorties non banales de ce circuit en fonction des valeurs de ses entrées.

▶ Une mémoire morte de $2^p$ mots de $q$ bits peut être considérée comme un circuit combinatoire comportant $p$ entrées notées $(a_0, \ldots, a_{p-1})$ et $q$ sorties notées $(s_0, \ldots, s_{q-1})$. Lorsque l'on présente sur les entrées de ce circuit un nombre $A$ compris entre 0 et $2^p - 1$ inclus, on obtient sur les sorties le mot dont l'adresse est $A$.

▶ Quelques rappels culturels : un octet est un mot de 8 bits ; 1 kilo-octet (kO) regroupe 1024 octets, et 1 méga-octet (MO) regroupe 1024 kO.

***Question 9*** Expliquez comment on peut réaliser $\mathcal{CC}(n)$ avec une mémoire morte.

***Question 10*** Quelle est la taille en kO d'une mémoire morte utilisée pour réaliser $\mathcal{CC}(8)$ ?

***Question 11*** Quelle est la plus grande valeur de $n$ pour laquelle on peut réaliser $\mathcal{CC}(n)$ avec une mémoire morte de 4 MO ? Peut-on envisager d'utiliser une mémoire morte pour réaliser $\mathcal{CC}(32)$ ?

# 3 Découpage d'un paragraphe en lignes

▶ On se propose de décrire plusieurs algorithmes de découpage d'un paragraphe en lignes et de justification de celles-ci, en vue de leur impression ou de leur affichage sur un écran. Un paragraphe sera fourni sous forme d'une chaîne de caractères (type `string`) ; on rappelle qu'en Caml, l'indexation des caractères d'une chaîne commence à 0 : si `s` est de type `string`, la longueur de `s` est donnée par `string_length s` ; le premier caractère est donné par `s.[0]` et le dernier par `s.[string_length s - 1]`.

▶ L'emploi de références est interdit. Le temps d'exécution de chaque fonction devra être un grand $\mathcal{O}$ de la taille de ses arguments ; par exemple, le découpage en lignes d'un paragraphe devra s'effectuer en un temps linéaire par rapport à sa longueur. Lorsque l'on vous demande d'*imprimer*, vous devrez utiliser les fonctions `print_string: string -> unit` et/ou `print_newline: unit -> unit`.

▶ Un paragraphe est formé de *mots* séparés par des *espaces* : un mot est une suite de caractères ne contenant aucun espace, qui commence au début du texte ou est précédée d'au moins un espace, et se termine à la fin du texte ou est suivie d'au moins un espace. Par exemple, le paragraphe suivant (encadré pour faciliter la compréhension) :

```
Ceci n'est qu'un essai.
```

comporte cinq mots, de longueurs respectives 4, 5, 5 et 6 (le point compte dans le dernier mot). On suppose pour l'instant que tous les caractères ont la même largeur, égale à celle d'un espace.

***Question 1*** Écrire une fonction :

```
décompose_en_mots: string -> string list
```

spécifiée comme suit : `décompose_en_mots p` décompose le paragraphe $p$ en mots. Par exemple :

```
décompose_en_mots "Ceci n'est qu'un essai.";;
```

rendra la liste `["Ceci";"n'est";"qu'un";"essai."]`.

▶ Le premier algorithme à mettre en œuvre est l'algorithme *glouton* : on place sur chaque ligne le maximum de mots consécutifs ; lorsqu'un mot ne peut tenir sur la ligne courante, on va à la ligne.

***Question 2*** Écrire une fonction :

```
prend_une_ligne: int -> string list -> string list*string_list
```

spécifiée comme suit : `prend_une_ligne n l` décompose la liste de mots $\ell = (m_1, m_2, \ldots, m_q)$ en un couple $(\ell_1, \ell_2)$ défini comme suit : soit $p$ le plus grand indice vérifiant $\sum_{1 \leqslant i \leqslant p} |m_i| \leqslant n - p + 1$ ; alors $\ell_1 = (m_1, \ldots, m_p)$ et $\ell_2 = (m_{p+1}, \ldots, m_q)$ (étant entendu que $\ell_2$ est la liste vide si $p = q$). Par exemple :

```
prend_une_ligne 12 ["Ceci";"n'est";"qu'un";"petit";"essai."];;
```

rendra le couple `(["Ceci";"n'est"],["qu'un";"petit";"essai."])`.

***Question 3*** Écrire une fonction :

```
découpe_en_lignes: int -> string -> string list list
```

spécifiée comme suit : `découpe_en_lignes n p` découpe le paragraphe $p$ en lignes de largeur au plus égale à $n$. Par exemple :

```
découpe_en_lignes 13 "Ceci n'est qu'un tout petit essai.";;
```

rendra la liste `[["Ceci";"n'est"];["qu'un";"tout"];["petit";"essai."]]`.

***Question 4*** Donner une condition nécessaire et suffisante portant sur $p$ et $n$ pour que ce découpage soit possible.

***Question 5*** Écrire une fonction :

```
imprime_ligne: string list -> unit
```

spécifiée comme suit : `imprime_ligne l` imprime les mots qui constituent la liste $\ell$, en insérant un espace exactement entre deux mots consécutifs.

***Question 6*** Écrire une fonction :

```
imprime_texte: int -> string -> unit
```

spécifiée comme suit : `imprime_texte n p` imprime le paragraphe $p$, après l'avoir découpé en lignes pouvant tenir dans une largeur $n$ au moyen de l'algorithme glouton.

▶ Le résultat fourni par `découpe_en_lignes` n'est pas satisfaisant, esthétiquement : le bord droit du texte est irrégulier. On va donc lui appliquer une *justification*, opération qui consiste à ajouter au besoin des espaces supplémentaires entre les mots de chaque ligne pour amener sa largeur totale à la valeur $n$ exactement. L'objectif est l'écriture d'une fonction :

```
justifie_ligne: int -> string list -> string list
```

spécifiée comme suit : `justifie_ligne n l` justifie la ligne $\ell$, en répartissant équitablement des espaces supplémentaires entre les mots qui la composent, pour amener sa longueur à $n$. Par exemple :

```
justifie_ligne 31 ["Ceci";"est";"un";"autre";"essai."];;
```

rendra la liste de chaînes suivante, dans laquelle on a matérialisé les espaces entre mots des étoiles :

```
["Ceci";"***";"est";"***";"un";"***";"autre";"**";"essai."]
```

Nous utiliserons le mot *insert* pour parler d'une suite d'espaces consécutifs séparant deux mots ; ainsi l'exemple précédent utilise quatre inserts, dont les trois premiers ont une largeur égale à 3 et le dernier a une largeur égale à 2.

▶ Soit $\ell = (m_1, m_2, \ldots, m_q)$ une ligne (présentée comme liste de mots) que l'on souhaite justifier. Pour des raisons esthétiques, on veut que les espaces insérés entre les mots de cette ligne soient répartis le plus régulièrement possible ; plus précisément, les largeurs des inserts doivent, soit être toutes égales à une même valeur $k$, soit ne prendre que deux valeurs distinctes $k$ et $k+1$

***Question 7*** Exprimer $k$ en fonction de $n$, du nombre $q$ de mots de la ligne, et de la somme $v = \sum_{1 \leqslant i \leqslant q} |m_i|$ des longueurs de ces mots.

***Question 8*** Écrire une fonction

```
répartit_espaces: int -> string list -> string list
```

spécifiée comme suit : `répartit_espaces n l` construit une liste $(e_1, e_2, \ldots, e_{q-1})$ d'inserts permettant de justifier la ligne $\ell = (m_1, m_2, \ldots, m_q)$ dans une largeur $n$.

***Question 9*** Écrire alors la fonction

```
justifie_ligne: int -> string list -> string list
```

spécifiée comme suit : `justifie_ligne n l` construit la liste $(m_1, e_1, m_2, e_2, \ldots, e_{q-1}, m_q)$ représentant la ligne justifiée.

***Question 10*** Écrire une fonction

```
imprime_texte_justifié: int -> string -> unit
```

spécifiée comme suit : `imprime_texte_justifié n p` découpe le paragraphe $p$ en lignes de longueur au plus $n$, les justifie et les affiche.

***Question 11*** Soit $(m_i)_{1 \leqslant i \leqslant Q}$ la liste des mots d'un paragraphe $p$ ; on note $x_i$ la longueur du mot $m_i$. Donner une condition *suffisante* portant sur $n$ et sur la suite $(x_i)_{1 \leqslant i \leqslant Q}$ pour que l'impression justifiée soit réalisable au moyen de l'algorithme que l'on vient de mettre en œuvre.

***Question 12*** Quel problème pose, en général, la dernière ligne d'un paragraphe ainsi composé ? Comment proposez-vous de le régler ?

**FIN**