Concours canadien de mathématiques

Une activité du Centre d'éducation en mathématiques et en informatique
Université de Waterloo, Waterloo (Ontario)

Concours canadien d'informatique

pour les bourses de SYBASE®

Le mardi 2 mars 1999

Problème 1
Jeu de cartes

Fichier d'entrée : card.in Fichier de sortie : card.out

Rédiger un programme qui tient compte des points marqués par deux joueurs d'un jeu de cartes simple. Le jeu est composé de 52 cartes, soit quatre cartes de chaque valeur.

En français	En anglais
deux	two
trois	three
quatre	four
cinq	five
six	six
sept	seven
huit	eight
neuf	nine
dix	ten
valet	jack
dame	queen
roi	king
as	ace

Les valets, les dames, les rois et les as sont considérés comme des cartes à valeur élevée.

Les cartes sont battues et le jeu de cartes est placé sur la table, faces en dessous. Le joueur A retourne la carte du dessus du jeu et la place sur un tas. Le joueur B retourne à son tour la carte du dessus du jeu et la place sur le tas. Les joueurs A et B jouent à tour de rôle jusqu'à ce qu'il n'y ait plus de cartes dans le jeu. Les points se calculent comme suit.

- si un joueur retourne un as, alors qu'il reste au moins quatre cartes à retourner dans le jeu et si aucune de ces quatre cartes n'est une carte à valeur élevée, il marque quatre points;
- si un joueur retourne un roi, alors qu'il reste au moins trois cartes à retourner dans le jeu et si aucune de ces trois cartes n'est une carte à valeur élevée, il marque trois points;
- si un joueur retourne une dame, alors qu'il reste au moins deux cartes à retourner dans le jeu et si aucune de ces deux cartes n'est une carte à valeur élevée, il marque deux points;
- si un joueur retourne un valet, alors qu'il reste au moins une carte à retourner dans le jeu et si celle-ci n'est pas une carte à valeur élevée, il marque un point.

Le fichier d'entrée contiendra 52 lignes. Chaque ligne contiendra la valeur (en minuscules) d'une carte, en anglais. La première ligne indique la première carte à retourner; la ligne suivante, la prochaine carte à retourner, et ainsi de suite. Chaque fois qu'un joueur compte des points, imprimez la ligne.

`Joueur` *`X`* `compte` *`n`* `point(s).`

où *X* représente le nom du joueur (A ou B) et *n*, le nombre de points marqués (1, 2, 3 ou 4). À la fin de la partie, imprimez le total des points de chaque joueur sur deux lignes.

`Joueur A :` *`n`* `point(s).`
`Joueur B :` *`m`* `point(s).`

Données d’essai

```
three
seven
queen
eight
five
ten
king
eight
jack
queen
six
queen
jack
eight
seven
three
ten
four
king
nine
six
seven
ace
four
jack
ace
ten
nine
ten
queen
ace
king
seven
two
five
two
five
nine
three
king
six
eight
jack
six
five
four
two
ace
four
three
two
nine
```

Résultat des données d’essai

```
Joueur A compte 2 point(s).
Joueur A compte 1 point(s).
```

```
Joueur A compte 3 point(s).
Joueur B compte 3 point(s).
Joueur A compte 1 point(s).
Joueur B compte 4 point(s).
Joueur A : 7 point(s).
Joueur B : 7 point(s).
```

Problème 2
L'an 2000

Fichier d'entrée : y2k.in Fichier de sortie : y2k.out

Vous saviez que le concours présenterait un problème sur l'an 2000. Eh bien le voici.

On vous remet un document contenant du texte et des données numériques, dont possiblement des dates. Vous devez relever les années représentées par deux chiffres et réimprimer le document tout en représentant ces années par quatre chiffres. Vous pouvez supposer que toute année représentée par un nombre inférieur ou égal à 24 renvoie aux années 2000 et que toute année représentée par un nombre supérieur ou égal à 25 correspond aux années 1900 (p. ex., 16 représente l'année 2016, alors que 26 correspond à l'année 1926). *Oui, nous savons que cette règle laisse entendre que votre grand-mère n'est peut-être pas encore née.*

Votre programme devra relever les dates présentées sous l'une des trois formes suivantes :

```
jj/mm/aa
aa.mm.jj
Month,jj,aa
```

où *jj* est un nombre de deux chiffres, de 01 à 31, représentant la journée; *mm* est un nombre de deux chiffres, de 01 à 12, représentant le mois; *aa* est un nombre de deux chiffres, de 00 à 99, représentant l'année; *month* est l'un des mois suivants (en anglais) January, February, March, April, May, June, July, August, September, October, November ou December. Les deux premières formes ne contiennent pas d'espace, mais la troisième contient un espace simple après *Month* et un autre après la virgule.

Les éléments d'une date devraient figurer sur la même ligne. Le programme ne doit pas tenir compte des dates réparties sur deux lignes ou de celles qui sont juxtaposées à une lettre de l'alphabet ou à un nombre. Les dates qui n'existent pas, telles que le 30 février 99, n'ont pas à être vérifiées.

La première ligne de données d'essai de votre programme contiendra un entier positif, *n*, indiquant le nombre de lignes de texte qui suit. Pour chaque ligne de texte, vous devez relever toutes les dates figurant sous l'une des formes décrites plus haut, tout en remplaçant les années représentées par deux chiffres par les années représentées par quatre chiffres, selon les directives mentionnées ci-dessus.

Données d'essai

```
4
Before 02/03/04, but not after December 19, 99,
there was a rehash of the 55.34.02 meeting.  A date, like November 15,
95 cannot traverse two lines, nor can it be surrounded by alphabetics
or numerics like this:  78November 01, 88, or 6801/12/03, or 02/03/04x.
```

Résultat des données d'essai

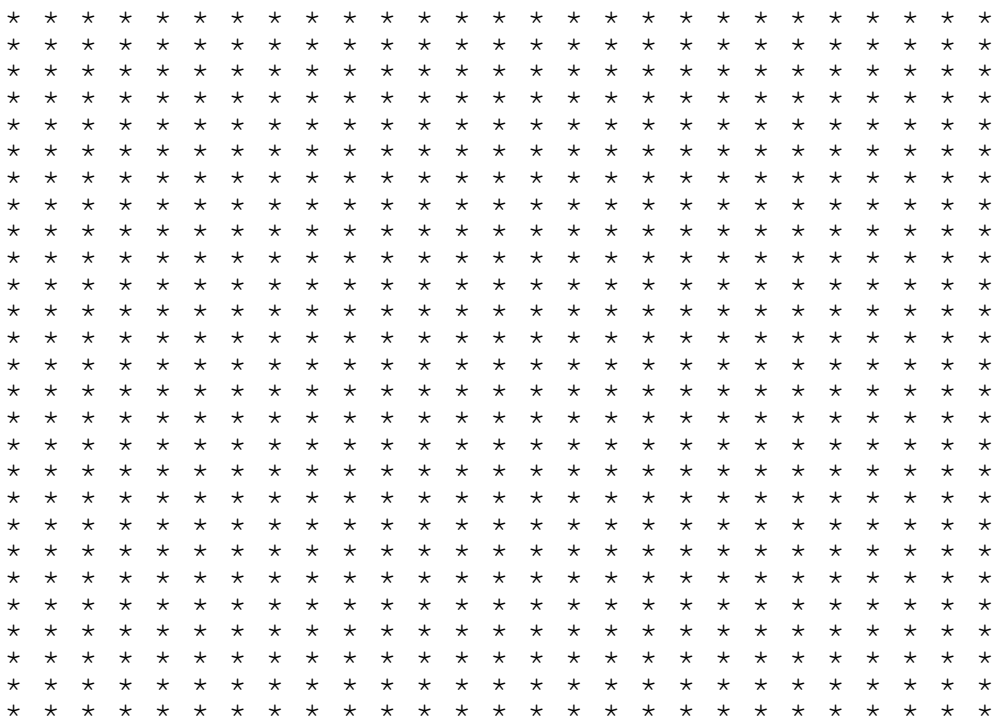
```
Before 02/03/2004, but not after December 19, 1999,
there was a rehash of the 55.34.02 meeting.  A date, like November 15,
95 cannot traverse two lines, nor can it be surrounded by alphabetics
or numerics like this:  78November 01, 88, or 6801/12/03, or 02/03/04x.
```

Problème 3
Fractales fragmentées

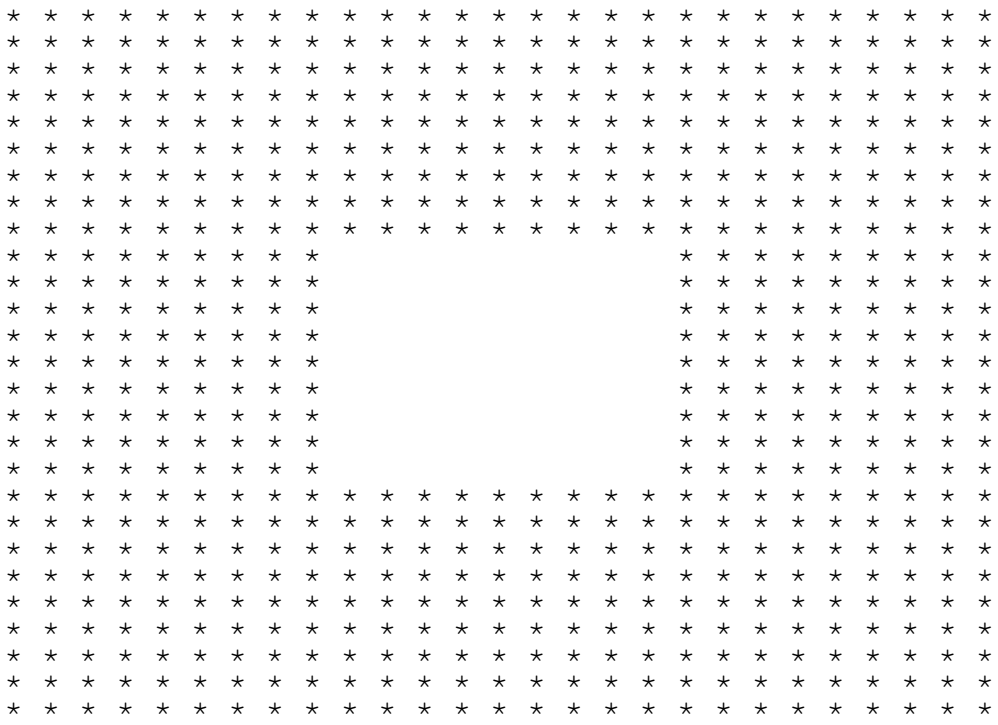
Fichier d'entrée : frac.in Fichier de sortie : frac.out

Une fractale est une figure géométrique semblable à ses parties lorsqu'on les observe à des échelles de plus en plus fines. Dans le présent cas, il est question d'une fractale précise dont nous ferons l'approximation par l'itération d'un processus de fragmentation.

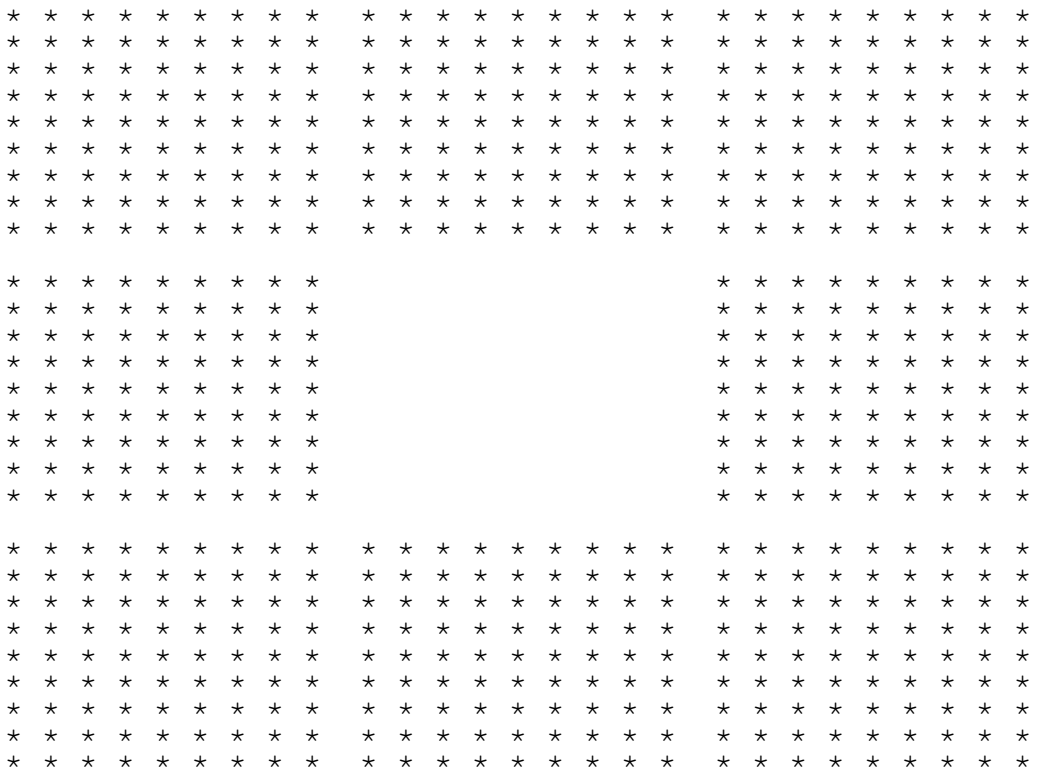
On commence par un carré plein dont la longueur des côtés est égale à 1. Par exemple□



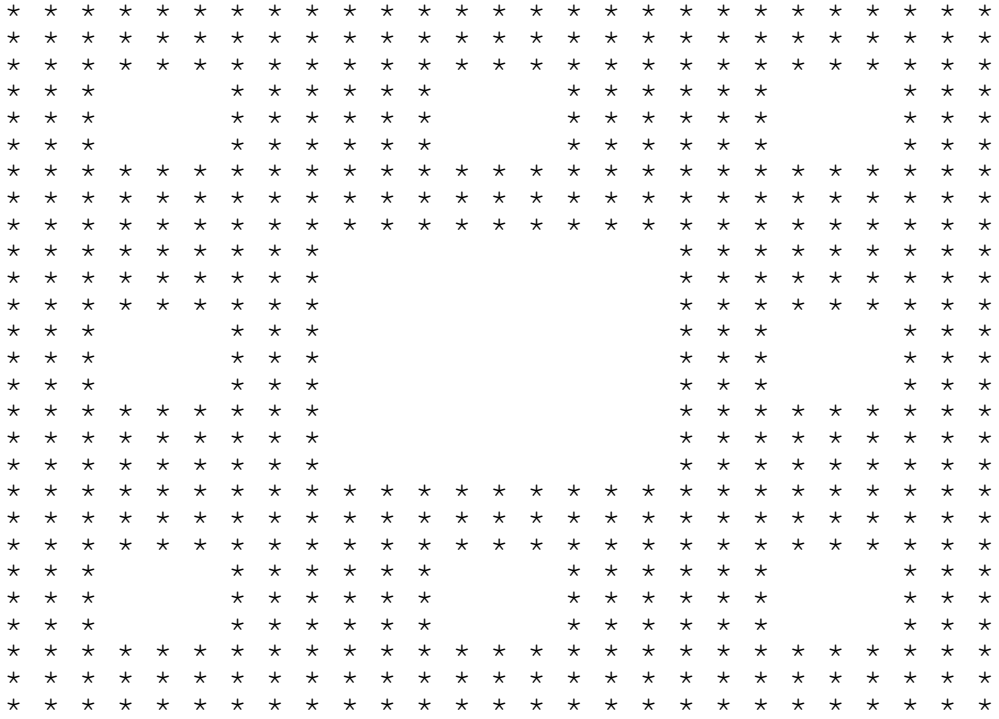
On retire du centre un carré dont la longueur des côtés est égale à 1/3□



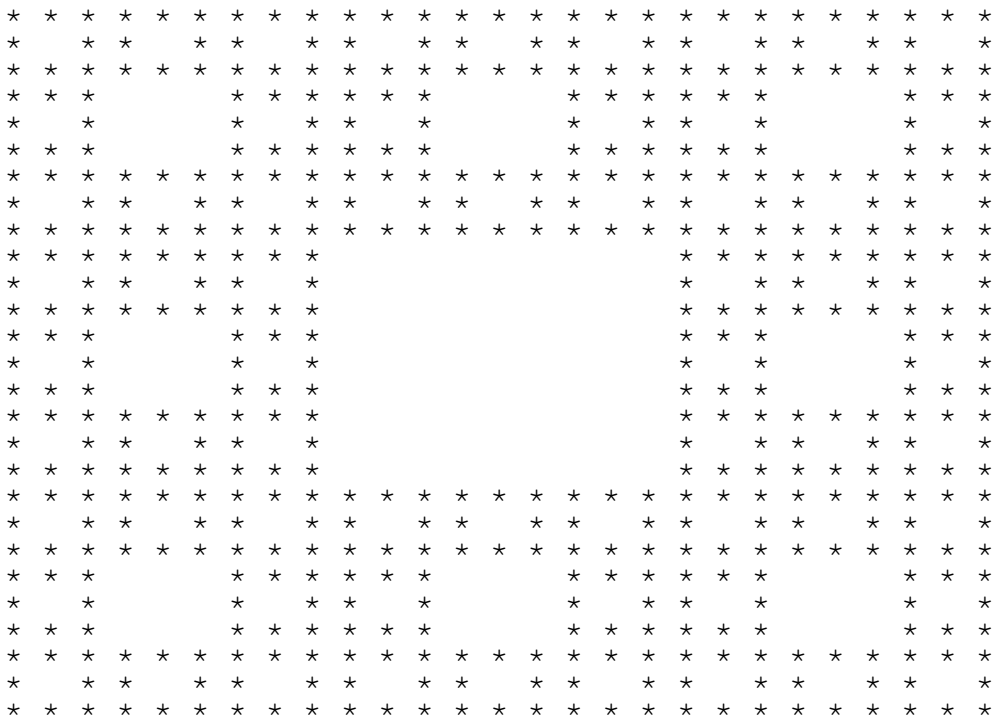
Vous remarquerez que cette figure est formée de huit carrés dont la longueur des côtés est égale à 1/3, comme dans le diagramme ci-dessous. *L'espace entre les carrés n'est présenté qu'à titre indicatif et ne figure pas dans la fractale*.



On fait subir le processus de fragmentation à chacun des carrés. Ainsi, après deux itérations (répétitions), la figure ressemble à⊡



Chacun des huit carrés est maintenant une copie réduite du carré troué obtenu après la première itération. Chacun de ces carrés contient huit carrés pleins, pour un total de 64 petits carrés. On fait subir le processus de fragmentation à chacun de ces carrés. Après trois itérations, voici la figure obtenue :



On obtient la vraie fractale après un nombre infini d'itérations. Comme on pourrait s'y attendre, chacune des huit sections de cette fractale est une copie conforme de la fractale entière, à une échelle de $\frac{1}{3}$.

Question 3 a)

Si le processus est répété *n* fois $(n \geq 1)$, combien de trous y aura-t-il dans le carré?

Question 3 b)

Après *n* itérations, quelle est l'aire totale de la partie remplie?

Question 3 c)

Après un nombre infini d'itérations, quelle est l'aire totale de la partie remplie?

Question 3 d)

Rédiger un programme qui calculera la fonction précisée après *n* itérations $(n \leq 5)$. Pour y arriver, représenter la figure sur une grille de 3^n sur 3^n, avec '*' pour indiquer une partie remplie et ' ' pour indiquer une partie vide. La figure sera trop large pour l'imprimer sur un écran simple ou sur une feuille de papier. Toutefois, les données d'essai permettront de produire une petite portion rectangulaire de la figure à imprimer.

La première ligne de données d'essai contient un entier positif *d*, indiquant le nombre d'ensembles de données d'essai qui suit. Chaque ensemble de données comporte cinq lignes indiquant :

- *n*, le nombre d'itérations $(0 \leq n \leq 5)$
- *b*, le numéro de la rangée du bas du rectangle à imprimer $\left(1 \leq b \leq 3^n\right)$
- *h*, le numéro de la rangée du haut du rectangle à imprimer $\left(b \leq h \leq 3^n\right)$
- *g*, le numéro de la colonne de gauche du rectangle à imprimer $\left(1 \leq g \leq 3^n\right)$
- *d*, le numéro de la colonne de droite du rectangle à imprimer $\left(g \leq d \leq 3^n\right)$

Veuillez noter que les rangées sont numérotées de bas en haut alors que les colonnes le sont de gauche à droite.

Calculez la forme de la figure et imprimez la portion rectangulaire demandée, avec une ligne de données de sortie par rangée. Imprimez tout d'abord la rangée du haut et terminez par celle du bas. Pour donner une apparence carrée au résultat, laissez en blanc une ligne horizontale entre les rangées (comme dans l'exemple ci-dessus). Insérez une ligne blanche dans les résultats après chaque jeu d'essai.

Données d'essai

```
1
3
2
10
5
27
```

Résultat des données d'essai

```
* * * * *                         * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * *
  * * *   * * *   * * *   * * *   * * *   * * *
* * * * * * * * * * * * * * * * * * * * * * * * *
    * * * * * * * * *       * * * * * * *       * * *
    *   * * *   * *       *   * * *   * *       *
    * * * * * * * * *       * * * * * * *       * * *
* * * * * * * * * * * * * * * * * * * * * * * * *
  * * *   * * *   * * *   * * *   * * *   * * *
```

Problème 4
Poursuite chevaleresque

Fichier d'entrée : knight.in Fichier de sortie : knight.out

Aux échecs, chaque pièce se déplace d'une façon particulière sur un échiquier de 8 cases sur 8 cases. Le but du jeu consiste à s'emparer des pièces de l'adversaire en arrivant sur leurs cases et finalement à s'emparer du roi.

Dans notre version du jeu, nous utiliserons un échiquier de dimensions variables sur lequel ne reposent que deux pièces : un pion blanc, qui se déplace inexorablement vers la rangée du haut de l'échiquier, une case à la fois, et un cavalier noir, qui peut se déplacer depuis sa position actuelle de l'une des huit façons suivantes : deux cases situées au-dessus ou en dessous *et* une case à gauche ou à droite; une case au-dessus ou en dessous *et* deux cases à gauche ou à droite. Le cavalier doit demeurer sur l'échiquier en tout temps; tout déplacement qui l'entraînerait à l'extérieur de l'échiquier est donc interdit. Dans le diagramme présenté ci-dessous, la position du cavalier est désignée par K et ses positions possibles, après un déplacement, sont représentées par les chiffres de 1 à 8.

```
. . . . . . .
. . 8 . 1 . .
. 7 . . . 2 .
. . . K . . .
. 6 . . . 3 .
. . 5 . 4 . .
. . . . . . .
```

Le pion se déplace en premier; puis, le cavalier et le pion se déplacent à tour de rôle. Le cavalier essaie d'occuper soit la case sur laquelle se trouve le pion (une *victoire*) ou la case située directement au-dessus du pion (*pat* sur le pion, c.-à-d. impasse pour ce dernier). Si le pion atteint la rangée du haut de l'échiquier, le cavalier perd aussitôt la partie (une *défaite*).

Rédiger un programme permettant de déterminer si le cavalier peut remporter une victoire, et le cas échéant, le nombre de déplacements qu'il lui faut pour y arriver. Si le cavalier ne peut pas gagner, votre programme doit déterminer s'il peut faire pat sur son adversaire et, le cas échéant, le nombre de déplacements qu'il lui faut pour y arriver. En dernier lieu, si le cavalier ne peut pas remporter une victoire ou faire pat sur son adversaire, votre programme devra être en mesure de calculer le nombre de déplacements que le cavalier fera avant que le pion ne remporte la victoire.

La première ligne de données d'essai contient un entier positif, *n*, soit le nombre de parties à analyser. Chaque partie comprend six lignes de données d'essai contenant :

- *r*, le nombre de rangées de l'échiquier $(3 \le r < 100)$
- *c*, le nombre de colonnes de l'échiquier $(2 \le c < 100)$
- *pr*, le numéro de la rangée correspondant à la position de départ du pion $(1 \le pr \le r)$
- *pc*, le numéro de la colonne correspondant à la position de départ du pion $(1 \le pc \le c)$
- *kr*, le numéro de la rangée correspondant à la position de départ du cavalier $(1 \le kr \le r)$
- *kc*, le numéro de la colonne correspondant à la position de départ du cavalier $(1 \le kc \le c)$

Le pion et le cavalier auront deux positions de départ distinctes. La rangée 1 se situe au bas de l'échiquier et la rangée *r*, au haut. La colonne 1 se situe à gauche et la colonne *c,* à droite.

Données d’essai

```
3
99
99
33
33
33
35
3
3
1
1
2
3
99
99
96
23
99
1
```

Résultat des données d’essai

```
Victoire en 1 déplacement(s) du cavalier.
Pat (impasse) en 1 déplacement(s) du cavalier.
Défaite en deux déplacement(s) du cavalier.
```

Problème 5
Arithmétique à l'aide de lettres

Fichier d'entrée : letter.in Fichier de sortie : letter.out

Un type populaire de jeu de crayon consiste à utiliser des lettres pour représenter des chiffres dans un énoncé mathématique. Par exemple,

```
 SEND
+MORE
-----
MONEY
```

qui représente :

```
 9567
+1085
-----
10652
```

Votre tâche consiste à lire des séries de trois « mots » et d'attribuer un chiffre particulier à chaque lettre de façon à ce que le troisième mot corresponde à la somme des deux premiers.

Le fichier d'entrée commence par une ligne comportant un entier positif, *n*, représentant le nombre de séries de données que contient le fichier. Chaque série de données se compose de trois lignes comportant chacune un mot, où le troisième mot correspond à la somme des deux premiers. Les mots ne contiendront pas plus de 20 caractères en haut de casse (majuscules).

Le fichier de sortie consiste en *n* séries de trois lignes contenant chacune la représentation numérique de chaque mot de la série de données. Chaque série de données ne doit comporter qu'une seule solution. Laissez une ligne blanche après la sortie de chaque jeu d'essai.

Données d'essai

```
2
SEND
MORE
MONEY
MEND
COPE
CONEY
```

Résultat des données d'essai

```
9567
1085
10652

9567
1085
10652
```