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 3 mars 1998

Problème A
Censure

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

La Société de prévention des blasphèmes sur Internet a constaté une croissance du nombre de groupes de discussion sur le W3 (World Wide Web). Un groupe de discussion permet aux utilisateurs du W3 de rédiger des lignes de texte pouvant être transmises à tous les autres utilisateurs. La Société est préoccupée par le nombre de mots de quatre lettres transmis dans le cadre de ces groupes de discussion et a proposé l'utilisation obligatoire d'un logiciel permettant d'éliminer tous les mots de quatre lettres que contient chaque message. Votre travail consiste à rédiger le contenu de ce logiciel.

Les données en entrée de votre programme comprennent un nombre entier, *n*, seul sur une ligne, suivi de *n* lignes de texte. Chaque ligne de texte contient des mots séparés d'espaces. Chaque mot est composé de lettres de l'alphabet. Les mots adjacents ne sont séparés que d'un seul espace. Il n'y a pas d'espace avant le premier mot, ni après le dernier mot de chaque ligne. Les lignes ne comptent pas plus de 80 caractères.

Les données de sortie de votre programme doivent comprendre *n* lignes de texte, dans lesquelles chaque mot de quatre lettres doit être remplacé par quatre astérisques. Les lignes doivent être séparées par une ligne vierge.

Exemple d'un fichier des données

```
2
The quick brown fox jumps over the lazy dog
Now is the time for all good people to come to the aid of the party
```

Exemple du fichier des résultats

```
The quick brown fox jumps **** the **** dog

Now is the **** for all **** people to **** to the aid of the party
```

Problème B

Jeu de « Nombres-croisés »

Fichier d'entrée : rien

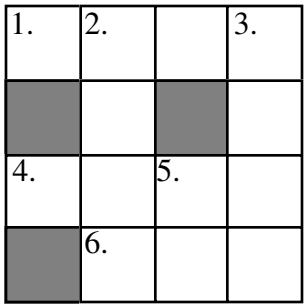
Fichier de sortie - partie 1 : perfect.out
Fichier de sortie - partie 2 : cube.out

Solutionner les trois problèmes suivants. Les solutions aux deux premiers problèmes doivent être obtenues à l'aide d'un ordinateur. La solution au troisième problème peut être trouvée sans l'aide d'un ordinateur, mais doit être inscrite à la grille faisant partie du formulaire de renseignements et de réponses.

1. Rédigez un programme permettant d'imprimer les nombres parfaits compris entre 1000 et 9999 inclusivement. Un nombre parfait est un nombre entier positif égal à la somme de ses propres diviseurs. Par propre diviseur, on entend tout diviseur inférieur au nombre lui-même. Par exemple, 6 est un nombre parfait, car 1 + 2 + 3 = 6. Les nombres doivent être imprimés un par ligne.

2. Rédigez un programme permettant de fournir tous les nombres entiers compris entre 100 et 999 inclusivement, égaux à la somme du cube de leurs chiffres. Les nombres doivent être imprimés un par ligne.

3. Utilisez le résultat de la partie 1 et 2, et de tout autre programme que vous avez rédigé pour remplir le jeu de « nombres-croisés » suivant.



Indices :
HORIZONTAL :
1. Un nombre parfait de quatre chiffres
4. Un palindrome de quatre chiffres. (Un palindrome est un nombre qui peut être lu indifféremment de gauche à droite et de droite à gauche. Par exemple : 1221)
6. Un nombre de trois chiffres égal à la somme des cubes de ses chiffres

VERTICAL :
2. Une paire de nombres premiers ayant un écart de deux entre eux (par exemple : 3 et 5)
3. Un carré parfait de quatre chiffres
5. Un nombre premier

Problème C

Rover sur Mars

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

Un rover (petit véhicule tout-terrain télécommandé permettant d'explorer une planète) se déplace sur une surface parfaitement plane (sans obstacle) de la planète Mars. Le rover est en communication radio avec la sonde spatiale qui l'a transporté. La sonde accumule et transmet au rover les instructions qu'elle reçoit de la Terre. Le rover fait plusieurs excursions. Chacune d'elles a comme point de départ le module de descente de la sonde orienté vers une direction connue. À la fin de chaque excursion, le module de descente doit calculer et transmettre une séquence d'instructions permettant de ramener le rover à son point de départ.

Le rover exécute une séquence d'instructions transmises par le module de descente. Chaque instruction lui indique de franchir un multiple d'une unité de distance exacte ou de tourner exactement de 90 degrés vers la gauche ou la droite.

L'instruction « avancer » est codée sur deux lignes consécutives dans le fichier des données en entrée. La première ligne contient le nombre premier 1, alors que la seconde contient un nombre premier non négatif *n*, soit la distance à franchir.
L'instruction « tourner à droite » est codée sur une seule ligne d'entrée comprenant le nombre premier 2.
L'instruction « tourner à gauche » est codée sur une seule ligne d'entrée comprenant le nombre premier 3.
Par exemple, la séquence d'instructions suivante indique au rover de tourner à gauche, puis d'avancer sur une distance de 10 unités, de tourner à droite et enfin d'avancer sur une distance de 5 unités.

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

Votre travail consiste, en fonction d'une telle séquence d'instructions, à déterminer la distance que doit franchir le rover pour retourner au module de descente et la séquence d'instructions la plus courte qui permettra le retour du rover à ce même point. Dans l'exemple ci-dessus, le véhicule doit franchir 15 unités de distance pour revenir, et la séquence d'instructions la plus courte est :

```
2
1
10
2
1
5
```

Le fichier des données en entrée commence par une ligne contenant un nombre premier positif, *n*, soit le nombre d'excursions du rover. Les instructions d'excursion occupent les lignes subséquentes du fichier des données en entrée. Chaque excursion consiste en un certain nombre d'instructions suivies d'une ligne contenant 0. Les données en entrée ne comportent pas d'erreur ou de ligne vierge. Le rover se déplace à moins de 10 000 unités de distance au cours de chaque excursion.

Suite à la page suivante

Pour chaque excursion, le fichier des données en entrée doit comprendre la ligne :

La distance est K

où K représente la distance (en unités) que le rover doit franchir pour retourner au module de descente. Les lignes suivantes doivent indiquer la plus courte séquence d'instructions pour ramener le rover au module de descente. Une ligne vierge doit séparer les lignes de résultat des différentes excursions.

Remarque : des points seront accordés pour les distances ne comprenant pas le retour au module de descente.

Exemple d'un fichier des données

```
3
2
3
3
0
3
1
10
2
1
5
0
1
5
2
1
5
3
3
1
1
0
```

Exemple du fichier des résultats

```
La distance est 0

La distance est 15
2
1
10
2
1
5

La distance est 9
1
4
3
1
5
```

Problème D
Loterie

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

Vous venez de gagner à la loterie. Il vous suffit de répondre correctement à la question d'habileté suivante pour remporter un gros lot de plusieurs millions :

1234 + 4567 x 11

Durant vos vingt secondes de réflexion, vous voyez votre fortune vous échapper, car vous hésitez entre les réponses suivantes :

(1234 + 4567) x 11 = 63811

ou

1234 + (4567 x 11) = 51471

Finalement, vous répondez 63811, mais c'est la mauvaise réponse. Votre professeur de mathématiques a préparé cette question en espérant que vous vous souviendriez que les multiplications sont effectuées avant les additions. La bonne réponse est 51471.

Votre travail consiste à rédiger un programme permettant d'insérer les parenthèses aux questions d'habileté de loterie comme celle posée ci-dessus afin d'indiquer clairement l'ordre des opérations.

Les données en entrée de votre programme consistent en une ligne comprenant un nombre premier, *n*, suivi de *n* lignes de données en entrée. Chacune des *n* lignes des données en entrée contient une expression composée de nombres premiers et des opérateurs +, - et X (X en majuscule) signifiant respectivement une addition, une soustraction et une multiplication. Les nombres premiers adjacents sont séparés d'un opérateur. Il y a un seul espace avant et après chaque opérateur, et les lignes de données en entrée ne doivent pas comporter plus de 80 caractères.

Votre résultat doit comprendre les mêmes *n* lignes, avec une ligne vierge entre chacune et des parenthèses insérées dans les *n* lignes pour indiquer l'ordre des opérations. Les multiplications doivent être effectuées en premier, de droite à gauche, et les additions et soustractions doivent être faites par la suite de gauche à droite. Les espaces situés avant et après les opérateurs doivent être conservés.

Exemple d'un fichier des données

```
3
10 + 20 X 30
1 + 2 + 3 - 4
123 + 456 X 789 - 876
```

Exemple du fichier des résultats

```
10 + (20 X 30)

((1 + 2) + 3) - 4

(123 + (456 X 789)) - 876
```

Problème E

Passage montagneux

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

Alp l'alpiniste se trouve sur la corniche nord-ouest d'un secteur carré en zone montagneuse. Il veut se frayer un passage jusqu'au coin opposé (sud-est). Alp est présentement à une altitude ne nécessitant pas d'apport en oxygène, mais il devra en consommer plus haut. Au besoin, la consommation d'oxygène se fait à un débit d'une unité par pas horizontal.

Le coin nord-ouest du terrain a pour coordonnées (1,1), alors que les coordonnées du coin sud-ouest sont (*n,n)*,où *n* est un nombre premier positif lu du fichier d'entrée. L'altitude de chaque point (*x,y*), ($1 \leq x, y \leq n$), est un nombre premier lu du fichier d'entrée; chaque niveau d'altitude occupe une ligne dans le fichier d'entrée.

Alp se déplace par pas horizontaux, où chaque pas le fait avancer d'une unité de distance vers le nord, le sud, l'est ou l'ouest. Alp doit rester dans la région carrée et ne peut monter ou descendre plus de 2 unités d'altitude par pas. Si l'altitude au début ou à la fin du pas requiert de l'oxygène, Alp consomme une unité d'oxygène durant son pas.

La première ligne de données en entrée est un nombre premier positif indiquant le nombre de déplacements que Alp doit effectuer. Chaque déplacement nécessite un certain nombre de lignes de données en entrée. La première ligne de chaque déplacement contient un nombre premier $n \leq 25$, indiquant la dimension de la zone carrée du terrain. Les prochaines n^2 lignes contiennent chacune un nombre premier représentant l'altitude d'un point précis du terrain. Les altitudes sont fournies pour les points selon l'ordre suivant :
(1, 1), (1, 2), (1, 3) ... (1, *n*), (2, 1), (2, 2), ... (*n*, 1), (*n*, 2), ... (*n*, *n*).

Pour chaque déplacement, advenant l'existence d'un passage, les données en entrée sont indiquées sur une ligne unique contenant un nombre premier représentant le nombre d'unités d'oxygène consommées. S'il n'existe pas de passage, le résultat est une ligne unique contenant le message « PASSAGE IMPOSSIBLE ». Les lignes de résultat des déplacements doivent être séparées d'une ligne vierge.

Suite à la page suivante

Exemple d'un fichier des données

```
2
5
5
4
3
2
1
7
5
6
6
6
8
8
8
9
6
9
6
9
9
6
4
5
4
5
3
2
4
9
9
4
```

Exemple du fichier des résultats

```
5

PASSAGE IMPOSSIBLE
```