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 Sun microsystems

Le mardi 25 février 2003

Commanditaires :

University of Waterloo

Sun microsystems

take it to the nth

Problème J1 - Le trident

Un *trident* se présente sous la forme d'une fourche à trois pointes (ou dents). En voici une représentation visuelle à partir d'astérisques espacés.

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

Dans l'exemple ci-contre, chacune des pointes est formée de trois astérisques, distancées l'une de l'autre par deux astérisques. Le manche est formé de quatre astérisques dans le prolongement de la pointe centrale.

Il est possible de concevoir des tridents de formes diverses en modifiant les trois paramètres suivants : t (la hauteur des pointes), s (l'espacement entre les pointes) et h (la longueur du manche). Dans l'exemple précité, ces paramètres sont les suivants : $t = 3$, $s = 2$ et $h = 4$.

Votre tâche consiste à concevoir un programme interactif qui permette d'imprimer un trident particulier à partir des trois paramètres *t, s* et *h*. Vous tenez pour acquis que la valeur de ces trois paramètres se trouve entre 0 et 10.

Exemple de session. *Mettre la saisie de l'utilisateur en italiques*

```
Entrez la hauteur des pointes:
   4
Entrez l'espacement entre les pointes:
   3
Entrez la longueur du manche:
   2
*    *    *
*    *    *
*    *    *
*    *    *
*********
     *
     *
```

Problème J2 - Des photos parfaitement agencées

Roy a devant lui une pile de photos tirées de l'annuaire des étudiants. Il souhaite recouvrir l'ensemble d'une surface plane de photos de manière à former un rectangle plein dont le périmètre soit le plus petit possible. Les photos devront toutes être faciles à visualiser et occuperont chacune une surface d'une unité sur unité.
Par exemple, il pourrait disposer douze photos selon la configuration suivante - chacune des photos est représentée par un `X`.

```
XXXX
XXXX
XXXX
```

Bien entendu, il pourrait en modifier l'organisation pour obtenir la configuration suivante :

```
XXX
XXX
XXX
XXX
```

qui aurait le même périmètre, soit 14 unités.

Il vous faudra axer les photos de manière interactive afin de vous permettre de calculer en permanence un entier relatif positif *C*, à savoir le nombre de photos qui constitueront la forme géométrique. Pour chacune des photos, on devrait imprimer le rectangle plein (qui recouvre toute la surface) et dont le périmètre sera le plus petit possible. De plus, vous devrez imprimer les dimensions du rectangle.

À toutes fins utiles, vous supposerez que vous disposez de moins de soixante-cinq mille photos. Le programme cessera de fonctionner lorsque *C* sera égal à zéro.

Exemple de session. *Mettre la saisie de l'utilisateur en italiques*

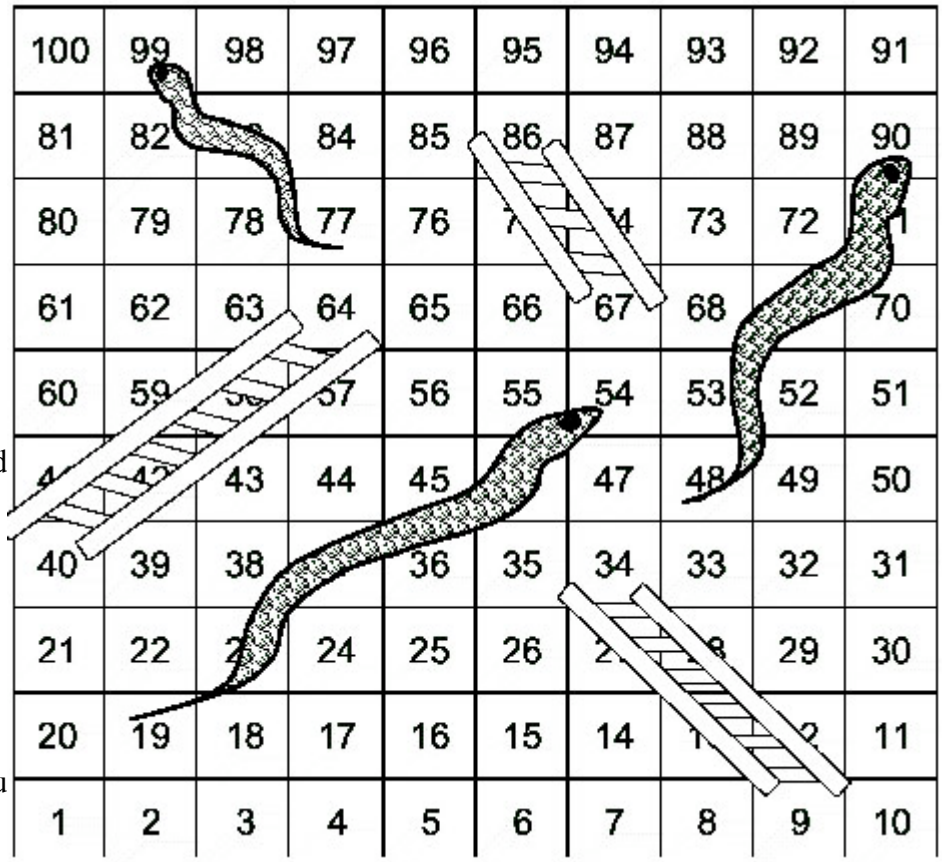
```
Inscrivez le nombre de photos :
   100
Le périmètre minimal est égal à 40 et aux dimensions suivantes :
10 x 10

Inscrivez le nombre de photos :
   15
Le périmètre minimal est égal à 40 et aux dimensions suivantes :
3 x 5
```

Inscrivez le nombre de photos :
195
Le périmètre minimal est égal à 40 et aux dimensions suivantes :
13 x 15

Inscrivez le nombre de photos :
0

Problème J3/S1 - Serpents et Échelles



L'illustration ci-contre représente une planche du jeu « Serpents et Échelles ». Dans ce jeu, les joueurs lancent les dés pour déterminer et faire avancer leur jeton en fonction du pointage obtenu. Ainsi, lorsque le jeton s'arrête sur une case qui comporte une échelle, le joueur se rend à la case où se trouve la partie supérieure de l'échelle. À l'inverse, lorsque le jeton s'arrête sur une case qui contient un serpent, le joueur descend à la case où se trouve la queue du serpent. Le but du jeu est de parvenir à dernière case de la planche. Nous devons préciser que, pour ce faire, le coup de dés doit correspondre exactement au nombre de cases qui séparent le jeton de la dernière case, autrement le jeton demeure sur la même case.

Pour vous aider à jouer à ce jeu au moyen d'un téléphone cellulaire lorsque vous êtes en voyage, vous concevrez un programme qui reproduit le mouvement du jeton sur la planche de jeu qui, bien entendu, est adapté à un ordinateur de poche. Ainsi, à chacun des coups de dés, le programme vous informera de la nouvelle position de votre jeton.

Dès l'activation du programme, vous occuperez la première case du jeu. Le programme déterminera à partir du pointage du coup de dés obtenu (dont la valeur est comprise entre 2 et 12) et indiquera la nouvelle position du jeton. De plus, le programme affichera le message « Vous avez gagné la partie » lorsque le jeton parvient à la dernière case et met fin à la partie. De même, lorsque l'un des joueurs inscrit le nombre « 0 » (et non un nombre compris entre 2 et 12), le programme imprimera le message « Fin de la partie! » et, effectivement, mettra fin à la partie.

Ainsi, vous vous servirez de la planche de jeu ci-dessus qui comporte trois serpents (des cases 54 à 19, 90 à 48 et 99 à 77) et trois échelles (des cases 9 à 34, 40 à 64, et 67 à 86).

Exemple de session. ***Mettre la saisie de l'utilisateur en italiques***

Inscrivez la somme du coup de dés :
9
Vous vous trouvez maintenant à la case 10.
Inscrivez la somme du coup de dés :
11
Vous vous trouvez maintenant à la case 21.
Inscrivez la somme du coup de dés :
12
Vous vous trouvez maintenant à la case 33.
Inscrivez la somme du coup de dés :
7
Vous vous trouvez maintenant à la case 64.
Inscrivez la somme du coup de dés :
3
Vous vous trouvez maintenant à la case 86.
Inscrivez la somme du coup de dés :
5
Vous vous trouvez maintenant à la case 91.
Inscrivez la somme du coup de dés :
10
Vous vous trouvez maintenant à la case 91.
Inscrivez la somme du coup de dés :
9
Vous vous trouvez maintenant à la case 100.
Vous avez gagné la partie!

Problème J4/S2 - Un peu de poésie

Fichier d'entrée: **poetry.in**
Fichier de sortie: **poetry.out**

Un poème, sous sa forme la plus simple, est composé de quatre vers formés d'un ou de plusieurs mots en lettres minuscules ou majuscules (ou les deux) et espacés les uns des autres.

La dernière syllabe d'un mot formé d'une séquence de lettres se terminera par une voyelle (« a », « e », « i », « o » ou « u » excluant « y »). Si le mot terminant le vers ne comporte pas de voyelle, la dernière syllabe formera le mot. Deux vers riment lorsque leur dernière syllabe est identique, sans tenir compte de la casse.

Vous organiserez les rimes de chacun des vers selon les catégories suivantes :

- les rimes parfaites : les quatre vers riment
- les rimes suivies : 1er et 2^{e} vers rimés et 3^{e} et 4^{e} vers rimés
- les rimes croisées : 1er et 3^{e} vers rimés et 2^{e} et 4^{e} vers rimés
- les rimes embrassées : 1er et 4^{e} vers rimés et 2^{e} et 3^{e} vers rimés
- les rimes libres : toute autre forme de vers rimés

La première ligne du programme contiendra un nombre relatif *N,* soit le nombre de vers du poème, $1 \leq N \leq 5$. Les autres 4*N* vers inscrits contiennent les vers du poème. Chacun des vers est composé d'au plus quatre-vingt (80) caractères (lettres ou espaces) comme on l'a indiqué précédemment.

Le programme permettra d'obtenir un poème composé d'un nombre *N* de vers lesquels devraient comporter les termes « parfaites », « suivies », « croisées », « embrassées » ou « libres » qui désignent le type de rimes du vers.

Notez: Il n'y a pas d'accents dans le texte, et il ne faut pas les mettre dans la sortie.

Exemple d'entrée 1

```
1
voici Caroline
elle aime nager
elle va a la piscine
apres manger
```

Sortie pour l'exemple d'entrée 1

```
croisées
```

Exemple d’entrée 2

2
il faut pouvoir
et il faut vouloir
vous allez voir
ils vont devoir
en hiver ils pensent
aux plages
et aux nuages
Ils attendent

Sortie pour l’exemple d’entrée 2

parfaites
embrassées

Exemple d’entrée 3

2
Nous cherchons
Et nous recherchons
Pour trouver la reponse
On pense
comment ca va
comme ci comme ca
comment ca va
ca va bien

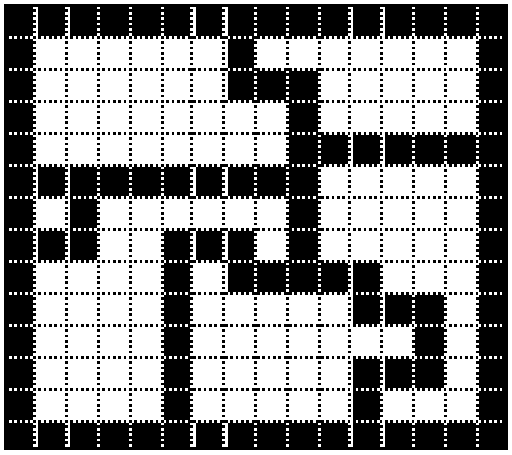
Sortie pour l’exemple d’entrée 3

suivies
libres

Problème J5/S3 – Plan d'un étage

Fichier d'entrée: **floor.in**
Fichier de sortie: **floor.out**

Le plan ci-contre montre la disposition des pièces d'un étage de la maison séparées par un mur. On peut reproduire ce plan sur une grille à l'aide des caractères « I » et « . » (qui représentent respectivement les murs et l'aire des pièces; les entrées ne figurent pas au plan) qui occupent chacun une surface d'un mètre carré.



Le diagramme ci-dessus nous indique que cet étage de la maison comporte six pièces.

On vous remet le plan de l'étage et des lattes de parquet en bois dur. Vous déterminez le nombre de pièces où on installera ce type de recouvrement (vous commencerez par les pièces les plus grandes, pour ensuite suivre un ordre décroissant). Vous devez installer ce recouvrement dans toutes les pièces jusqu'à l'épuisement des lattes de bois. Le programme devrait permettre de calculer le nombre de pièces où il sera possible d'installer ce type de recouvrement et la surface de recouvrement (en mètres carrés) à couvrir une fois l'approvisionnement en lattes épuisé. Les pièces auront une surface maximale de 64 mètres carrés.

La première ligne du programme contiendra l'étendue de la surface de recouvrement en mètres carrés. La deuxième ligne précisera un nombre relatif r – de 1 à 25 - qui représente le nombre de rangées de la grille. La troisième ligne contiendra le nombre relatif c - de 1 à 25 - qui représente le nombre de colonnes de la grille. Le nombre r de lignes restantes contiennent un nombre c de caractères de la grille de données.

Exemple d’entrée 1

```
105
14
16
IIIIIIIIIIIIIIII
I......I.......I
I......III.....I
I........I.....I
I........IIIIIII
IIIIIIIII.....I
I.I......I.....I
III..III.I.....I
I....I.IIIII...I
I....I.....III.I
I....I.......I.I
I....I.....III.I
I....I.....I...I
IIIIIIIIIIIIIIII
```

Sortie pour l’exemple d’entrée 1

```
4 chambres, 1 mètre(s) carré(s) qui reste(nt)
```

Exemple d’entrée 2

```
13
2
3
.I.
.I.
```

Sortie pour l’exemple d’entrée 2

```
2 chambres, 9 mètre(s) carré(s) qui reste(nt)
```

Problème S4 – Les sous-chaînes

Fichier d'entrée: **substr.in**
Fichier de sortie: **substr.out**

Combien de sous-chaînes peut-on obtenir à partir d'une chaîne *S* donnée?

Par exemple, si *S* = « abc », *S* possède sept sous-chaînes distinctes : {« », « a », « b », « c », « ab », « bc » et « abc » }. Nota : la sous-chaîne vide et *S* sont considérées comme des sous-chaînes de *S.*

D'autre part, si *S* = « aaa », alors *S* possède uniquement quatre sous-chaînes distinctes : {« »,« a »,« aa »,« aaa »}.

La première ligne du programme permet de calculer *N*, soit le nombre de tests élémentaires; à chacun de ces derniers succède une ligne qui indique la valeur de *S*, une chaîne composée entre un et mille caractères alphanumériques. Votre programme produira une ligne pour chacun des tests élémentaires et permettra de calculer le nombre de sous-chaînes distinctes de *S*. Dans la mesure du possible, efforcez-vous de concevoir un programme efficace.

Exemple d'entrée

```
2
abc
aaa
```

Sortie pour l'exemple d'entrée

```
7
4
```

Problème S5 – Les aléas de la vente de camions lourds

Fichier d'entrée: **truck.in**
Fichier de sortie: **truck.out**

Vous vendez des camions poids lourd servant notamment au transport d'autres camions. Vous devez conduire l'un de ces camions poids lourds dans un vaste territoire marécageux; vous devrez donc traverser des ponts. En fait, on retrouve un pont sur chacune des routes interurbaines. Il n'existe pas de route directe entre deux villes. Ces ponts peuvent supporter une charge utile égale à un nombre entier relatif compris entre zéro et cent mille.

On vous a remis une liste des villes (ou *villes destinations*) où vous devez rencontrer des clients intéressés par le genre de camions que vous vendez. Au moment de choisir le véhicule que vous conduirez, vous devez résoudre le problème suivant : quel est le poids maximal du camion qui me permettrait de me rendre sans danger à destination? Votre tâche est de concevoir un programme qui permettrait de résoudre le problème.

La première ligne du programme comportera trois nombres entiers relatifs : *c*, *r* et *d* qui précisent le nombre total de villes, de routes interurbaines, de *villes destinations*. On attribue aux villes un nombre entre 1 à *c*. On compte au plus 1 000 villes et 10 000 routes. Un nombre *r* d'autres lignes précise le triplet *x y* w qui indique que la route passe par les villes *x* et *y* et qu'elle peut supporter une charge utile maximale de *w*. Le nombre *d* des lignes restantes servira à énumérer les villes destinations (la liste comportera au moins une ville) de votre parcours.

La première ville constituera votre point de départ et ne sera pas considérée comme une ville destination. Vous vous rendrez dans un nombre *d* de villes destinations dans l'ordre que vous choisirez, mais vous devez cependant visiter toutes ces villes. Le programme vous permettra de calculer un nombre entier relatif unique qui représente le poids maximal du véhicule que vous conduirez dans l'ensemble du nombre *d* de villes destination

Exemple d'entrée

```
5 7 3
1 2 20
1 3 50
1 4 70
1 5 90
2 3 30
3 4 40
4 5 60
2
4
5
```

Sortie pour l'exemple d'entrée

```
30
```