Concours
canadien de
mathématiques

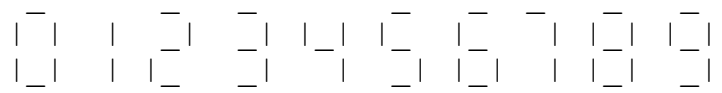
Une activité du Centre d'éducation
en mathématiques et en informatique
Université de Waterloo, Waterloo (Ontario)

Concours canadien d'informatique

pour les bourses de 

Le mardi 26 février 2002

Problème J1



La plupart des dispositifs numériques affichent les nombres à l'aide d'un dispositif d'affichage à sept segments. Ces sept segments sont disposés ainsi :

Pour ce problème, chaque segment est représenté par trois astérisques disposés en ligne, tel qu'illustré ci-dessus. Tout chiffre de 0 à 9 peut être affiché par l'illumination des segments appropriés. Par exemple, pour afficher le chiffre 1, il s'agit d'illuminer les deux segments du côté droit :

Pour afficher le chiffre 4, il faut illuminer quatre segments :

Écrivez un programme qui acceptera une seule saisie de chiffre par l'utilisateur, puis qui affichera ce chiffre à l'aide du dispositif d'affichage à sept segments. Vous pouvez supposer que chaque segment comprend trois astérisques.

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

```
Entrez un chiffre entre 0 et 9 :
9
        * * *
      *       *
      *       *
      *       *
        * * *
              *
              *
              *
        * * *
```

Problème J2
AmeriCanadian

Les Américains et les Canadiens anglais n'orthographient pas tous les mots de la même façon. Les Américains écrivent «□eighbor□ et «□olor□ tandis que les Canadiens écrivent «□eighbour□ et «□olour□.

Écrivez un programme qui permet de changer la graphie américaine pour la graphie canadienne.

Votre programme doit interagir avec l'utilisateur de la façon suivante. L'utilisateur doit taper un mot (maximum de 64 lettres). Si le mot est épelé selon l'orthographe américaine, le programme doit donner l'épellation canadienne de ce mot. Si le mot ne semble pas être épelé à l'américaine, il doit être retourné sans changement. Si l'utilisateur tape «□quitter!□, le programme doit s'arrêter.

Les règles de détection de l'orthographe américaine sont très simples□si un mot a plus de quatre lettres et que son suffixe est constitué d'une consonne suivie de «□r□, vous pouvez conclure qu'il est épelé à l'américaine et que pour convertir le mot selon l'orthographe canadienne, la particule «□r□ sera remplacée par «□ur□.
Note: La lettre "y" doit être traiteé comme une voyelle.

Répondre par une saisie clavier et une sortie écran.

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

```
Entrez les mots à traduire□:
color
colour
for
for
taylor
taylour
quitter!
```

Problème S1J3
Le déjeuner du conseil étudiant

Le conseil étudiant de votre école organise un déjeuner de bienfaisance. Comme les étudiants les plus âgés sont à la fois les plus sages et les plus fortunés, les membres du conseil décident que le prix de chaque billet sera déterminé par le nombre d'années que l'étudiant fréquente l'école. Ainsi, un étudiant de première année doit acheter un billet ROSE, un étudiant de deuxième année achètera un billet VERT, un étudiant de troisième année doit acheter un billet ROUGE et un étudiant de quatrième année, un billet ORANGE.

Supposons que tous les billets se vendent. Chaque billet n'a qu'un prix. Entrez le coût d'un billet ROSE, d'un billet VERT, d'un billet ROUGE et d'un billet ORANGE (dans cet ordre précis). Entrez le montant d'argent exact à recueillir par la vente de billets. Donnez toutes les combinaisons de billets qui équivalent au montant à recueillir. Les combinaisons peuvent être affichées dans n'importe quel ordre. Donnez le nombre total de combinaisons trouvées. Donnez aussi le plus petit nombre de billets à imprimer pour recueillir le montant désiré, de façon à minimiser le coût d'impression.

Répondre par une saisie clavier et une sortie écran.

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

```
Coût des billets ROSES□:
1
Coût des billets VERTS□:
2
Coût des billets ROUGES□:
3
Coût des billets ORANGES□:
4
Combien doit être recueilli par la vente de billets?
3
Combinaisons possibles□:
Nombre de billets
ROSES□: 0         VERTS□: 0          ROUGES□: 1       ORANGES□: 0
ROSES□: 1         VERTS□: 1          ROUGES□: 0       ORANGES□: 0
ROSES□: 3         VERTS□: 0          ROUGES□: 0       ORANGES□: 0
Le nombre total de combinaisons est 3.
Le nombre minimal de billets à imprimer est 1.
```

Problème S2J4
Fractions

De nombreuses calculatrices perfectionnées ont une fonction qui permet de simplifier les fractions.

Vous devez écrire un programme qui acceptera un numérateur positif et un dénominateur positif comme saisie et qui réduira la fraction à sa plus simple forme, c'est-à-dire qu'elle ne pourra être réduite davantage et que le numérateur sera inférieur au dénominateur. Vous pouvez supposer que tous les numérateurs et dénominateurs constituent des fractions valides.

Répondre par une saisie clavier et une sortie écran.

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

```
numérateur
28
dénominateur
7
4

numérateur
13
dénominateur
5
2 3/5

numérateur
0
dénominateur
7
0

numérateur
55
dénominateur
10
5 1/2
```

Problème S3J5
Les yeux bandés

Rose et Colin jouent à un jeu dans leur cour. Comme la cour est rectangulaire, nous pouvons nous la représenter par une grille de *r* rangées et *c* colonnes. Rose et Colin placent des obstacles sur certains carrés.
Le jeu se joue comme suit :
Colin bande ses yeux, puis Rose l'amène vers certaines cases dans la cour. Elle l'oriente face au nord, au sud, à l'est ou à l'ouest. Colin ne connaît pas sa position initiale ni son orientation.
Rose donne ensuite comme consigne à Colin de faire une série de *m* déplacements dans la cour. Chaque déplacement se définit de l'une des façons suivantes :

F - avancer d'une case dans la direction à laquelle il fait face;

L - tourner de 90 degrés dans le sens contraire des aiguilles d'une montre, en restant dans le même carré;

R - tourner de 90 degrés dans le sens des aiguilles d'une montre, en restant dans le même carré.

Après avoir effectué ces déplacements, Colin se trouve dans une position finale. Il doit maintenant tenter de déterminer où il se trouve. Vous allez l'aider en écrivant un programme pour déterminer toutes les positions finales possibles. Supposez que la position initiale, la position finale et toutes les positions intermédiaires de Colin sont à l'intérieur de la cour mais jamais dans un carré qui contient un obstacle. Vous pouvez aussi supposer que Colin fait toujours face à une direction qui est parallèle aux limites de la cour (nord, sud, est ou ouest).

La saisie commence par *r* et *c* ($1 \le r \le 375$; $1 \le c \le 80$), chacun sur une ligne distincte. Les données suivantes sont les lignes *r* de caractères *c* décrivant la cour : un point marque une case que Colin peut traverser et le caractère X indique une case contenant un obstacle. Au-dessous de la grille se trouve le nombre *m* ($0 \le m \le 30000$) suivi de *m* lignes décrivant les déplacements de Colin. Chaque ligne ne contient qu'un caractère, soit F, L ou R.

Votre programme doit permettre de définir la grille de la cour en indiquant toutes les positions finales possibles au moyen du caractère *.

Exemple d'entrée (fichier d'entrée : blind.in)

```
2
4
....
.XX.
3
F
R
F
```

Sortie pour l'exemple d'entrée (fichier de sortie: blind.out)

```
.*..
.XX*
```

Problème S4
La traversée de pont

Un pont de corde traverse une gorge profonde. Il est possible de traverser le pont en groupes, mais il y a une limite (M) à la taille du groupe. Le temps pris par un groupe pour traverser est déterminé par la personne la plus lente. Vous êtes responsable de la sécurité sur le pont et votre rôle consiste entre autres à diriger la traversée des groupes. Les gens attendent en file; lorsqu'un groupe a traversé, vous informez les autres personnes que leur tour est venu de traverser. Les groupes peuvent être de différentes tailles mais un groupe ne peut contenir plus de M personnes. L'objectif est de faire traverser les gens le plus rapidement possible, tout en respectant l'ordre des personnes dans la file.

La première ligne de données contient un nombre entier M ($1 \le M \le 20$). La deuxième ligne contient Q ($1 \le Q \le 100$), soit la longueur de la file d'attente. Pour chaque personne qui attend, il y a deux lignes□la première contient le nom de la personne et la seconde est le temps pris par cette personne pour traverser le pont. **Rappelez-vous que le temps de traversée d'un groupe égale le temps maximal de traversée pris par une personne de ce groupe.**

Votre programme doit dresser la liste des noms des personnes de chaque groupe (un groupe par ligne), ce qui permettra d'obtenir le temps de traversée global minimal. Si quelques groupements obtiennent le même temps de traversée global, chacun d'eux peut être inclus dans la liste. La dernière ligne doit donner le temps de traversée total pour la file entière.

Exemple d'entrée (fichier d'entrée : bridge.in)

```
2
5
alice
1
bob
5
charlie
5
dobson
3
eric
3
```

Sortie pour l'exemple d'entrée (fichier de sortie: bridge.out)

```
Le temps total: 9
alice
bob charlie
dobson eric
```

Problème S4
La traversée de pont

Un pont de corde traverse une gorge profonde. Il est possible de traverser le pont en groupes, mais il y a une limite (M) à la taille du groupe. Le temps pris par un groupe pour traverser est déterminé par la personne la plus lente. Vous êtes responsable de la sécurité sur le pont et votre rôle consiste entre autres à diriger la traversée des groupes. Les gens attendent en file; lorsqu'un groupe a traversé, vous informez les autres personnes que leur tour est venu de traverser. Les groupes peuvent être de différentes tailles mais un groupe ne peut contenir plus de M personnes. L'objectif est de faire traverser les gens le plus rapidement possible, tout en respectant l'ordre des personnes dans la file.

La première ligne de données contient un nombre entier M ($1 \le M \le 20$). La deuxième ligne contient Q ($1 \le Q \le 100$), soit la longueur de la file d'attente. Pour chaque personne qui attend, il y a deux lignes□la première contient le nom de la personne et la seconde est le temps pris par cette personne pour traverser le pont. **Rappelez-vous que le temps de traversée d'un groupe égale le temps maximal de traversée pris par une personne de ce groupe.**

Votre programme doit dresser la liste des noms des personnes de chaque groupe (un groupe par ligne), ce qui permettra d'obtenir le temps de traversée global minimal. Si quelques groupements obtiennent le même temps de traversée global, chacun d'eux peut être inclus dans la liste. La dernière ligne doit donner le temps de traversée total pour la file entière.

Exemple d'entrée (fichier d'entrée : bridge.in)

```
2
5
alice
1
bob
5
charlie
5
dobson
3
eric
3
```

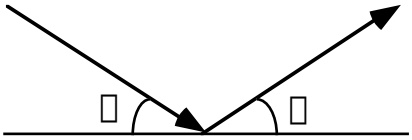
Sortie pour l'exemple d'entrée (fichier de sortie: bridge.out)

```
Le temps total: 9
alice
bob charlie
dobson eric
```

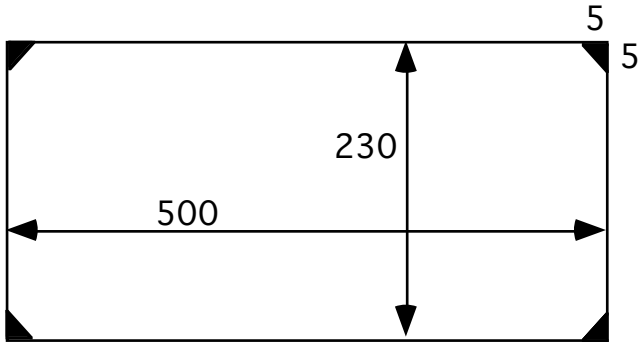
Problème S5
Suivez la balle bondissante!

Supposons que vous voulez créer un économiseur d'écran qui présente une balle bondissant sur les côtés. L'action se déroule jusqu'à ce que la balle frappe un coin (comme pour une table de billard). À ce moment, la balle disparaît de l'écran et celui-ci devient noir.

Les balles bondissent sur les murs selon les lois de la physique. L'angle de la balle approchant le mur (angle d'incidence) égale l'angle de la balle lorsqu'elle s'en éloigne (angle de réflexion).



Écrivez un programme qui déterminera le temps qui s'écoule avant que la balle bondissante ne disparaisse dans un coin, s'il y a lieu. Les dimensions de l'écran se situent dans l'intervalle de [100, 1000] unités. Considérez la balle comme un point de l'écran. Les «poches» dans les coins mesurent 5 unités le long de chaque mur et si une balle frappe le côté d'une «poche», elle continuera de bondir.
Voir l'exemple d'écran ci-dessous.



La saisie de données sera constituée de quatre nombres entiers, chacun sur une ligne distincte.

- n, m – la largeur et la hauteur de l'écran, $100 \le n, m \le 1000$
- p – la position initiale de la balle au bas de l'écran, $5 \le p \le n - 5$
- q – la position de la balle lorsqu'elle bondit du mur de droite, $5 \le q \le m - 5$

Le résultat attendu doit être un nombre entier qui indique le nombre de rebonds que la balle fait avant de disparaître dans un coin. Considérez le rebond du mur de droite comme étant le premier. Il est possible que la balle ne bondisse jamais dans un coin (en supposant une surface sans frottement). Dans ce cas, le résultat doit être 0. Vous n'avez pas à programmer cette animation.

Exemple d'entrée (fichier d'entrée : ball.in)

```
300
200
200
100
```

Sortie pour l'exemple d'entrée (fichier de sortie: ball.out)

```
2
```