Le Centre d'éducation en mathématiques et en informatique

Concours canadien d'informatique 2010

Niveau supérieur

Commanditaire :

University of Waterloo

Concours canadien d'informatique – Niveau supérieur

Règles et conseils à l'intention des participantes et des participants

1. Vous pouvez participer à un concours seulement. Pour participer au concours de niveau intermédiaire, il faut choisir l'autre trousse de problèmes.

2. Sur le formulaire **Information à l'intention des élèves**, indiquez que vous participez au concours de niveau **supérieur**.

3. Vous avez trois (3) heures pour accomplir le travail.

4. Vous pouvez prendre pour acquis que :
   - toutes les entrées se trouvent dans des fichiers nommés `sX.in`, X étant le numéro du problème ($1 \leq X \leq 5$).
   - toutes les sorties se font par l'écran.

   Puisque les entrées se trouvent dans des fichiers, il n'y aura aucune sollicitation. Les sorties doivent être IDENTIQUES à celles des exemples de sorties, par rapport à l'ordre, aux espaces, etc.

5. Vous devez faire votre propre travail. Les tricheurs seront punis sévèrement.

6. Il est interdit de faire appel à des caractéristiques auxquelles le juge, votre enseignant, n'a pas accès pendant l'évaluation de votre programme.

7. Vous pouvez consulter des livres et du matériel écrit. Tout matériel susceptible d'être lu électroniquement (par exemple un programme que vous avez écrit) est *interdit*. Cependant, vous pouvez faire appel aux bibliothèques reconnues pour vos langages de programmation : par exemple STL pour C++, java.util.*, java.io.*, etc. pour Java, et ainsi de suite.

8. Vous devez vous limiter aux applications de programmation ordinaires (éditeurs, compilateurs, débogueurs). Toutes les autres applications sont **interdites**. Leur utilisation entraînera une disqualification.

9. Utilisez des noms de fichier qui sont propres à chaque problème : par exemple, `s1.pas` ou `s1.c` ou `s1.java` (ou tout autre suffixe de fichier approprié) pour le problème S1. Ceci facilitera la tâche de l'évaluateur.

10. Votre programme sera exécuté avec des fichiers d'essai différents de ceux qui figurent comme exemples. Assurez-vous de vérifier votre programme au moyen d'autres fichiers d'essai. Pour certains problèmes, particulièrement les problèmes 4 et 5, des solutions peu performantes peuvent faire perdre des points. Assurez-vous d'avoir un code aussi performant que possible par rapport au temps.

11. Les deux premiers participants de chaque région du pays recevront une plaque et une somme de 100 $. Leur école recevra aussi une plaque. Les régions sont :
    - L'ouest (de la C.-B. au Manitoba)
    - Le nord et l'est de l'Ontatio
    - Toronto métropolitain
    - Le centre et l'ouest de l'Ontario
    - Le Québec et les provinces de l'Atlantique

12. Si vous vous placez parmi les 20 premiers participants et participantes dans le concours du niveau supérieur, vous serez invité à participer à l'Étape 2 du CCI, qui aura lieu à l'Université de Waterloo au mois de mai 2010. Si vous vous placez parmi les 4 premiers lors de l'Étape 2, vous serez invité à participer à l'équipe qui représentera le Canada à IOI 2010, au Canada. Notez que vous devez connaître C, C++ ou Pascal si vous êtes invité à l'Étape 2. Mais d'abord, vous devez réussir le concours d'aujourd'hui !

13. Consultez le site web du CCI à la fin du mois de mars pour connaître votre classement dans ce concours, pour voir comment on pouvait résoudre les problèmes et pour connaître le nom des gagnants. Voici l'adresse :

    `www.cemc.uwaterloo.ca/ccc`

Problème S1 : Achat d'un ordinateur

Description du problème
Pour améliorer votre rendement lors de l'ACI (Autre Concours d'Informatique), vous décidez d'acheter un autre ordinateur. Pour déterminer l'ordinateur qu'il faut acheter, vous prenez en compte les caractérisques suivantes :

– mémoire vive (en gigaoctets), que l'on notera M

– vitesse de l'unité centrale (en mégahertz), que l'on notera V

– espace sur le disque dur (en gigaoctets), que l'on notera D

D'après votre analyse, vous déterminez que l'ordinateur préféré est celui qui a la plus grande valeur de l'expression

$$2 * M + 3 * V + D.$$

Vous devez lire une liste donnée d'ordinateurs et sortir le nom des deux ordinateurs préférés, en ordre de préférence, soit le premier suivi du deuxième.

Précisions par rapport aux entrées
La première ligne d'entrée sera un entier n ($0 \leq n \leq 10000$). Chacune des n lignes suivantes contiendra les caractéristiques d'un ordinateur. Une caractéristique d'un ordinateur aura la forme suivante :

– nom de l'ordinateur, soit une chaîne de moins de 20 caractères

– la quantité de mémoire vive, soit un entier M ($1 \leq M \leq 128$)

– la vitesse de l'unité centrale, soit un entier V ($1 \leq V \leq 4000$)

– l'espace sur le disque dur, soit un entier D ($1 \leq D \leq 3000$)

Il y a une espace entre le nom, la valeur de M, la valeur de V et la valeur de D sur chaque ligne.

Précisions par rapport aux sorties
La sortie est le nom des deux ordinateurs préférés, un nom par ligne, en ordre décroissant de préférence. En cas d'égalité, choisissez l'ordinateur (ou les ordinateurs) dont le nom vient avant l'autre en ordre lexicographique (p. ex., « Apple » vient avant « Dell »). S'il y a un seul ordinateur dans la liste, sortez son nom sur une ligne (c.-à-d. ne pas l'imprimer deux fois).

Exemple d'entrée
```
4
ABC 13 22 1
DEF 10 20 30
GHI 11 2 2
JKL 20 20 20
```

Sortie pour l'exemple d'entrée

```
JKL
DEF
```

Explication de la sortie pour cet exemple d'entrée

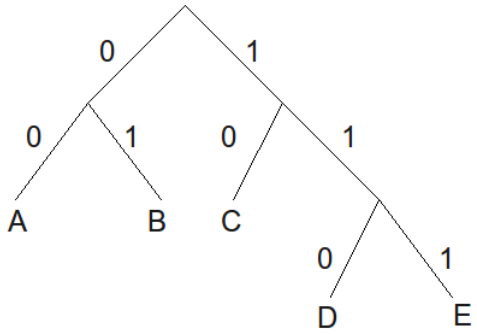
Pour l'ordinateur ABC, on a calculé une valeur de 93. Pour l'ordinateur DEF, on a calculé une valeur de 110. Pour l'ordinateur GHI, on a calculé une valeur de 30. Pour l'ordinateur JKL, on a calculé une valeur de 120. Donc, l'ordinateur JKL est le préféré, suivi de l'ordinateur DEF.

Problème S2 : Codage de Huffman

Description du problème
L'*encodage de Huffman*, créé par David Huffman en 1952, est un algorithme ingénieux employé dans la compression de données.

Il consiste à assigner une suite binaire particulière (un suite de 0 et de 1) à chaque caractère. Le code est *sans préfixe*, c'est-à-dire qu'une suite binaire qui représente un caractère n'est jamais le préfixe de la suite binaire d'un autre caractère.

Il faut mentionner que pour construire une suite binaire sans préfixe, on associe les caractères aux feuilles d'un arbre binaire. À partir de la racine, les chemins vers la gauche sont notés 0 et les chemins vers la droite sont notés 1. Le chemin depuis la racine jusqu'au noeud d'une feuille forme la suite binaire pour le caractère situé à ce noeud. Par exemple, l'arbre binaire suivant construit des suites binaires sans préfixe pour les caractères $\{$A, B, C, D, E$\}$:



Ainsi, le code pour A est 00, le code pour B est 01, le code pour C est 10, le code pour D est 110 et le code pour E est 111.

Un codage sans préfixe a pour avantage que n'importe quelle suite de codes peut être décodée de façon unique de manière à donner les caractères initiaux.

Vous devez lire un code Huffman (c'est-à-dire un ensemble de caractères et les suites binaires correspondantes), de même qu'une suite binaire, et décoder la suite binaire pour retrouver les caractères qu'elle représente.

Précisions par rapport aux entrées
La première ligne sera un entier k ($1 \leq k \leq 20$) qui représentera le nombre de caractères et de codes associés. Les k lignes suivantes contiennent un caractère, suivi d'une espace, suivie d'une suite binaire (d'une longueur qui ne dépassera pas 10) qui représente le code associé à ce caractère. Vous pouvez supposer que le caractère est un caractère de l'alphabet (c'est-à-dire de a à z et de A à Z). Vous pouvez supposer que la suite de codes binaires est sans préfixe. La $(k+2)^{\text{ième}}$ ligne contient le code qu'il faut décoder. Vous pouvez supposer que la suite binaire de ce code contient les codes associés aux caractères donnés et que la $(k+2)^{\text{ième}}$ ligne ne contient pas plus de 250 chiffres binaires.

Précisions par rapport aux sorties
Sur une ligne, sortez les caractères qui correspondent à la suite binaire donnée.

Exemple d'entrée
```
5
A 00
B 01
C 10
D 110
E 111
00000101111
```

Sortie pour l'exemple d'entrée
```
AABBE
```

Problème S3 : Boyau d'incendie

Description du problème
Dans votre quartier, il y a une rue plutôt exceptionnelle. Elle a la forme d'un cercle ayant une circonférence de 1 000 000. Il y a M ($1 \leq M \leq 1000$) maisons sur la rue. L'adresse de chaque maison correspond à la longueur d'arc depuis le point nord jusqu'à la maison, dans le sens des aiguilles d'une montre. L'adresse de la maison au point nord est 0.

Vous disposez de boyaux d'incendie qui peuvent être déployés le long de la rue. Or, vous voulez que le boyau le plus long dont vous aurez besoin ait une longueur minimale.

Vous devez placer k ($1 \leq k \leq 1000$) bouches d'incendie sur la rue de manière que la longueur maximale d'un boyau nécessaire pour relier une maison à une bouche d'incendie soit aussi petite que possible.

Précisions par rapport aux entrées
La première ligne sera un entier M, soit le nombre de maisons. Les M lignes suivantes contiennent chacun un entier qui représente l'adresse d'une maison particulière. Chaque adresse est un entier supérieur ou égal à 0 et inférieur à 1 000 000. La $(M+2)^{\text{ième}}$ ligne contient un entier k, soit le nombre de bouches d'incendie qu'il faut placer sur la rue. Il est possible de placer une bouche d'incendie devant une maison. Vous pouvez supposer que toutes les maisons ont des adresses différentes. Remarque : Au moins 40 % des points attribués à cette question correspondront à $M \leq 10$.

Précisions par rapport aux sorties
La sortie indiquera, sur une ligne, la longueur du boyau d'incendie qu'il faut de sorte qu'avec cette longueur, chaque maison puisse être désservie par la bouche d'incendie la plus près.

Exemple d'entrée
```
4
0
67000
68000
77000
2
```

Sortie pour l'exemple d'entrée
```
5000
```

Problème S4 : La ferme des animaux

Description du problème
Vous êtes responsable d'une ferme qui a N ($1 \leq N \leq 100$) animaux. Vous êtes allé au magasin où vous avec acheté $M = N$ enclos pre-fabriqués pour y loger les animaux. Les enclos satisfont aux conditions suivantes :

– Les enclos ont de 3 à 8 côtés.

– Si un côté est précisé pour deux enclos, c'est qu'il sert à séparer les deux enclos.

– Un côté qui n'est précisé qu'une fois relie l'enclos à l'extérieur.

– Au départ, il y a exactement un animal dans chaque enclos et aucun animal à l'extérieur.

Or, les animaux adorent jouer à un jeu qu'ils appellent « Sortons de l'enclos ». On attribue un coût à chaque côté de chaque enclos et ils déterminent le coût minimal qui permet à tous les animaux de se rencontrer au même endroit en écrasant des côtés de divers enclos. Le lieu de rencontre peut être à l'intérieur d'un enclos particulier ou à l'extérieur. De plus, lorsqu'un côté d'un enclos a été écrasé par un animal, tout autre animal peut y passer sans que le coût n'augmente.

Vous recevrez une description des enclos, de même que l'emplacement des animaux, et vous devrez déterminer le coût minimal qui permettrait aux animaux de se rencontrer au même endroit.

Précisions par rapport aux entrées
La première ligne sera un entier M, soit le nombre d'enclos. Les M lignes suivantes contiendront les descriptions des enclos, soit une description par ligne. Une description comportera trois éléments séparés d'une espace :

– Le premier élément est un entier c_e ($3 \leq c_e \leq 8$), qui décrit le nombre de côtés de l'enclos e.

– Le deuxième élément est une suite de c_e entiers qui décrivent les coins de chaque enclos, chaque entier étant inférieur ou égal à 1000.

– Le troisième élément est une suite de c_e entiers qui donnent le coût de chaque côté, chaque entier étant inférieur ou égal à 5000.

Les descriptions des coins et des coûts sont donnés en ordre cyclique. Par exemple, la description suivante d'un enclos

3 1 2 3 7 4 6

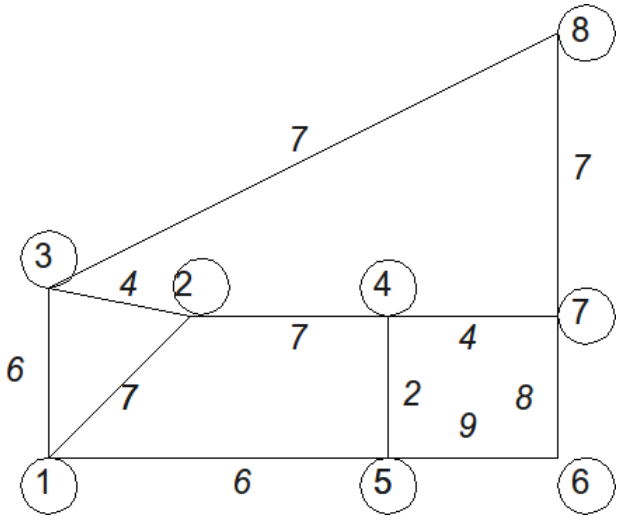
indique que l'enclos a 3 côtés, soit $(1,\ 2)$, $(2,\ 3)$ et $(3,\ 1)$ et que le côté $(1,\ 2)$ a un coût de 7, le côté $(2,\ 3)$ a un coût de 4 et le côté $(3,\ 1)$ a un coût de 6. Remarque : Au moins 20 % des points attribués à cette question correspondront à $N \leq 10$ et dans ces cas, aucun enclos n'aura plus de quatre côtés.

Précisions par rapport aux sorties
Sur une ligne, indiquez le coût minimal qui permettrait à tous les animaux de se retrouver au même endroit, soit dans un enclos ou à l'extérieur.

Exemple d'entrée
```
4
3 1 2 3 7 4 6
4 1 2 4 5 7 7 2 6
4 4 7 6 5 4 8 9 2
5 3 2 4 7 8 4 7 4 7 7
```

Sortie pour l'exemple d'entrée
```
10
```

Explication de la sortie pour cette entrée
La figure suivante représente les données de l'entrée.



Les nombres encerclés reprśentent les coins et les nombre en *italique* représentent les coûts des côtés. On remarque se si les côtés $(2,\ 3)$, $(4,\ 5)$ et $(4,\ 7)$ sont écrasés, tous les animaux peuvent se rencontrer dans l'enclos qui a cinq côtés.

Problème S5 : Arbre à nutriments

Description du problème
Il existe un arbre binaire à f feuilles $(1 \leq f \leq 50)$ dans lequel chaque feuille k produit une quantité n_k $(1 \leq n_k \leq 10\,000)$ de nutriments.

Les branches de cet arbre binaire (les lignes qui relient les noeuds de l'arbre) limitent la quantité de nutriments qui peuvent couler vers la racine de l'arbre. Vous avez X agents de croissance $(1 \leq X \leq 2500)$ qui peuvent être utilisés pour augmenter l'épaisseur d'une branche ou pour augmenter la production en nutriments d'une feuille. Au départ, chaque branche a une valeur de 1 et si vous lui donnez w agents de croissance, elle peut transporter $(1 + w)^2$ unités de nutriments. Si une feuille a une valeur initiale de n_k et qu'on augmente sa production de nutriments de s, sa production de nutriments atteint une valeur de $n_k + s$.

Lorsque deux branches se rencontrent, la quantité de nutriments qui coule est la somme des nutriments qui coulent dans les deux branches.

Vous devez déterminer la quantité maximale de nutriments que vous pouvez transporter à la racine.

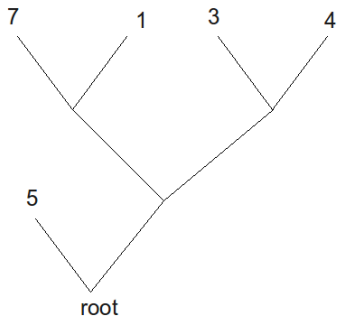
Précisions par rapport aux entrées
La première ligne d'entrée sera une description de l'arbre. La description peut être définie de façon récursive par un entier n_k $(1 \leq n_k \leq 10\,000)$ ou par $(A_G A_D)$, A_G et A_D étant les descriptions respectives des sous-arbres de gauche et de droite. La deuxième ligne d'entrée sera un entier X, soit le nombre d'agents de croissance que vous avez. Remarque : Au moins 30 % des points attribués à cette question correspondront à $f \leq 5$ et $X \leq 50$.

Précisions par rapport aux sorties
Sur une ligne, indiquez la quantié maximale de nutriments que vous pouvez transporter à la racine.

Exemple d'entrée
```
(5 ((7 1) (3 4)))
3
```

Sortie pour l'exemple d'entrée
```
7
```

Explication de la sortie pour cette entrée
Voici une représentation de la description de l'arbre :



Remarquer que si on attribue 2 agents de croissance à la branche gauche de la racine et 1 agent de croissance à la branche droite de la racine, il y aura 5 nutriments qui couleront de la feuille 5 vers la racine (puisque la branche a une capacité de $(1+2)^2$ unités, ou 9 unités de nutriments) et 2 nutriments qui couleront du sous-arbre droit vers la racine (puisque la branche qui relie le sous-arbre à la racine a une capacité de $(1+1)^2$ unités, ou 4 unités, mais elle est limitée à 2 unités, puisque les branches qui y sont reliées ont chacune une limite de 1).

D'autre part, si on attribuait 1 agent de croissance à la feuille de gauche dans le but d'augmenter sa production de nutriments de 5 unités à 6 unités et si on attribuait de 2 agents de croissance à la branche qui relie cette feuille à la racine (ce qui lui donnerait une capacité de $(1+2)^2$ unités, ou 9 unités), il y aurait 6 unités de nutriments qui se rendraient à la racine depuis la branche de gauche et 1 unité de nutriments qui s'y rendrait depuis le sous-arbre de droite.

Dans les deux cas, on peut transporter à la racine une quantité maximale de 7 unités de nutriments.