Le Centre d'éducation en mathématiques et en informatique

Concours canadien d'informatique 2009

Niveau intermédiaire

Commanditaire :

University of Waterloo

Concours canadien d'informatique – niveau intermédiaire

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 supérieur, 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 **intermédiaire**.

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

4. Vous pouvez prendre pour acquis que :
   – toutes les entrées se font par le biais du clavier ;
   – toutes les sorties se font par l'écran.

   Dans certains problèmes, on peut vous demander de fournir une sollicitation pour l'utilisateur. Si aucune sollicitation n'est requise, il n'est pas nécessaire d'en fournir une. 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, `j1.pas` ou `j1.c` ou `j1.java` (ou tout autre suffixe de fichier approprié) pour le problème J1. 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, 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 du niveau intermédiaire 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. 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 J1 : ISBN

Description du problème
Le numéro international normalisé du livre (numéro ISBN) est un code de 13 chiffres qui sert à identifier les livres. Chaque numéro a une propriété qui permet de déceler si le numéro a été écrit correctement.

Si on multiplie chaque chiffre du numéro alternativement par 1 et par 3, la somme des produits est appelée la somme-1-3. Cette somme doit toujours être un multiple de 10. Par exemple, pour calculer la somme-1-3 du numéro 9780921418948, on additionne

$$9*1+7*3+8*1+0*3+9*1+2*3+1*1+4*3+1*1+8*3+9*1+4*3+8*1$$

pour obtenir 120. Puisque la somme est un multiple de 10, le numéro est correct.

Vous devez écrire un programme qui calcule la somme-1-3 d'un numéro de 13 chiffres. Pour réduire le nombre de chiffres qu'il faut taper, vous pouvez supposer que les dix premiers chiffres seront toujours 9780921418, comme dans l'exemple ci-dessus. Votre programme devrait accepter comme entrée les trois derniers chiffres et calculer la somme-1-3 du numéro au complet. Utilisez le même format que celui des exemples suivants.

Exemple 1 d'une session interactive (entrées en *italique*)
```
Chiffre 11 ? 9
Chiffre 12 ? 4
Chiffre 13 ? 8
```

Sortie pour l'exemple 1
```
La somme-1-3 est 120
```

Exemple 2 d'une session interactive (entrées en *italique*)
```
Chiffre 11 ? 0
Chiffre 12 ? 5
Chiffre 13 ? 2
```

Sortie pour l'exemple 2
```
La somme-1-3 est 108
```

Problème J2 : La pêche

Description du problème
Il est nécessaire de bien gérer les espèces de poissons pour assurer leur revouvellement durable. Dans une rivière particulière, des limites ont donc été imposées sur la pêche de certaines espèces selon leur population. Ainsi on a associé un nombre de *points* à chaque espèce et le nombre total de points associés aux poissons attrapés ne doit pas dépasser une certaine limite.

Par exemple, supposons que chaque truite brune vaut 2 points, chaque brochet vaut 5 points et chaque doré vaut 2 points et qu'il y a une limite de 12 points pour la pêche sur une rivière. Il serait alors permis de prendre 3 truites brunes et 1 brochet. D'autres combinaisons seraient permises.

Vous devez écrire un programme qui prend pour entrée les points alloués pour les poissons d'une rivière et qui détermine le nombre de façons pour un pêcheur d'attraper *au moins* un poisson, tout en respectant les conditions.

Précisions par rapport aux entrées
L'entrée consistera en 4 entiers, soit un par ligne, représentant dans l'ordre les points pour les truites brunes, pour les brochets, pour les dorés et le total de points permis. Vous pouvez supposer que chaque entier sera supérieur à 0 et inférieur ou égal à 100.

Précisions par rapport aux sorties
Pour chaque combinaison de poissons qu'il est permis de prendre, une ligne indiquera dans l'ordre le nombre de truites brunes, de brochets et de dorés de la combinaison. Les combinaisons peuvent être énumérées dans n'importe quel ordre. La dernière ligne de la sortie indiquera le nombre total de façons différentes qu'il existe de pêcher des poissons, tout en respectant la limite établie.

Exemple d'entrée
```
1
2
3
2
```

Exemple de sortie
```
1 truite(s) brune(s), 0 brochet(s), 0 doré(s)
2 truite(s) brune(s), 0 brochet(s), 0 doré(s)
0 truite(s) brune(s), 1 brochet(s), 0 doré(s)
Nombre de façons d'attraper des poissons : 3
```

Problème J3 : À la bonne heure

Description du problème
Un service de téléphonie cellulaire à Ottawa émet l'heure de façon automatique à ses clients, de manière à refléter l'heure locale, peu importe où ils sont au Canada. Ceci assure que les messages affichent la bonne heure locale.

Par exemple, s'il est 1420 à Ottawa le mardi 24 février 2009 (l'heure est affichée selon le format militaire de 24 heures), l'heure correspondante à travers le Canada est donnée dans le tableau suivant :

H^{re} du Pacifique	H^{re} des Rocheuses	H^{re} du Centre	H^{re} de l'Est	H^{re} de l'Atlantique	H^{re} de Terre-Neuve
Victoria, BC	Edmonton, AB	Winnipeg, MB	Toronto, ON	Halifax, NS	St. John's, NL
Mardi	Mardi	Mardi	Mardi	Mardi	Mardi
24/2/2009	24/2/2009	24/2/2009	24/2/2009	24/2/2009	24/2/2009
1120 HNP	1220 HNR	1320 HNC	1420 HNE	1520 HNA	1550 HNT

Vous devez écrire un programme qui accepte l'heure à Ottawa, en format militaire de 24 heures, et qui donne l'heure dans chacune des villes ci-dessus, y compris à Ottawa. Vous pouvez supposer que l'heure de l'entrée sera valide (c'est-à-dire un entier de 0 à 2359 dont les deux derniers chiffres varieront de 00 à 59).

Remarquez que 2359 indique une minute avant minuit, que 0 indique minuit et que 13 indique 13 minutes après minuit. Il n'est pas nécessaire d'imprimer les zéros qui précèdent l'heure (on imprime 345 et non pas 0345) ; les entrées ne contiendront aucun zéro inutile.

Exemple d'entrée
```
1300
```

Exemple de sortie
```
1300 à Ottawa
1000 à Victoria
1100 à Edmonton
1200 à Winnipeg
1300 à Toronto
1400 à Halifax
1430 à St. John's
```

Problème J4 : Affichage

Description du problème
Suite aux succès remportés par l'équipe de ski, le conseil des élèves du Collège central d'informatique prépare des affiches électroniques qui afficheront `SKIEURS DU CCI CENT FOIS BRAVO` à divers endroits dans l'école. Selon son emplacement, une affiche est assez large pour contenir w caractères par ligne, y compris les espaces.

Voici comment les mots sont placés dans l'affiche. D'abord on place autant de mots que possible sur la première ligne, sans dépasser la limite de w caractères. Le premier mot d'une ligne commence dans la position la plus à gauche. Si une ligne contient plus d'un mot, le dernier mot doit se terminer dans la position la plus à droite. Des espaces supplémentaires sont insérées entre les mots de manière que les intervalles entre les mots soient aussi égaux que possible. Si les intervalles ne peuvent être égaux, les intervalles plus grands doivent paraître à la gauche des intervalles plus petits.

Votre programme lira la largeur w et présentera en sortie les lettres et les espaces de l'affiche sur écran. Utilisez le caractère « `.` » pour indiquer une espace.

Contrainte
Vous pouvez supposer que $w \geq 7$.

Exemple d'entrée 1 (entrée en *italique*)
`Entrer w :` *15*

Sortie pour l'exemple d'entrée 1
```
SKIEURS..DU.CCI
CENT.FOIS.BRAVO
```

Exemple d'entrée 2 (entrée en *italique*)
`Entrer w :` *26*

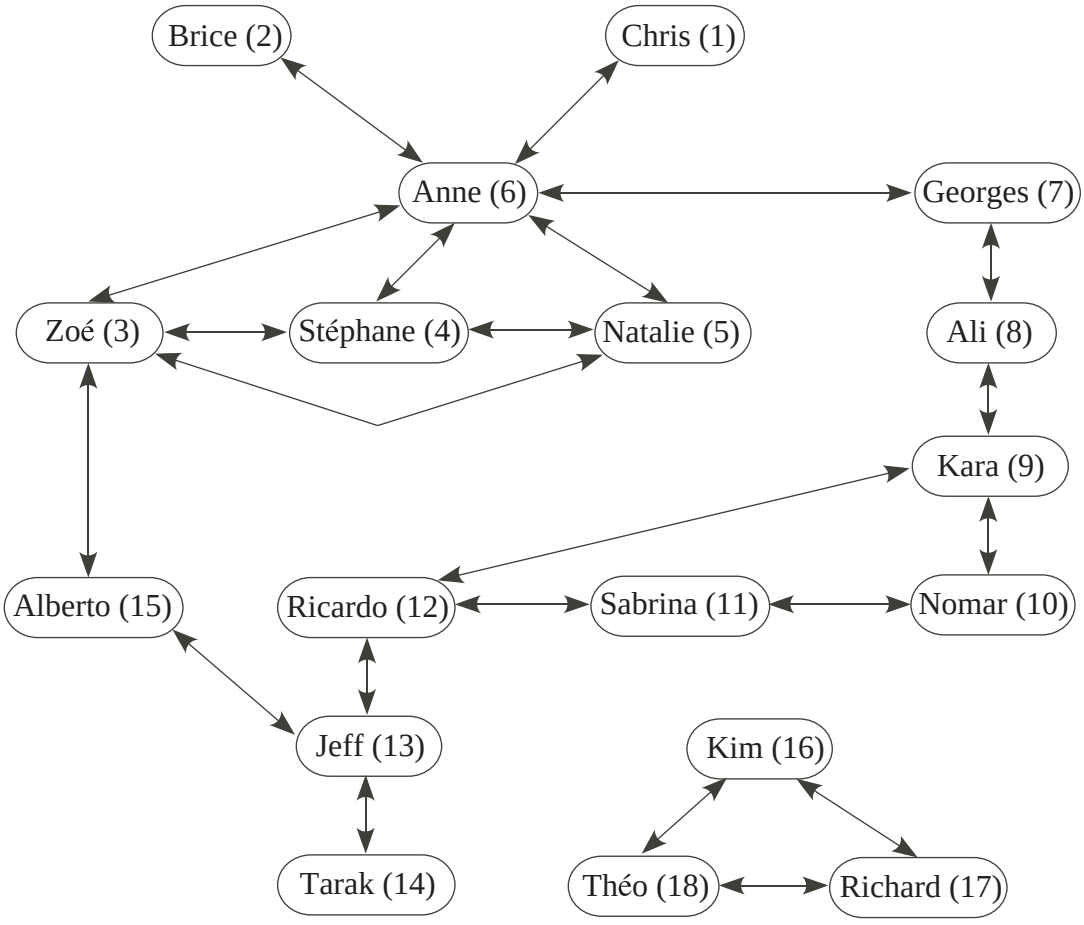
Sortie pour l'exemple d'entrée 2
```
SKIEURS..DU..CCI.CENT.FOIS
BRAVO......................
```

Problème J5 : Degrés de séparation

Description du problème

Facebook est sans doute le principal outil des élèves pour socialiser. Facebook suscite plusieurs questions reliées à l'informatique, telles que le degré de séparation deux personnes.

Par exemple, dans la figure ci-dessous, il y a plusieurs chemins qui relient Anne et Alberto. En voici quelques-unes :
– Anne → Zoé → Alberto
– Anne → Natalie → Zoé → Alberto
– Anne → Georges → Ali → Kara → Ricardo → Jeff → Alberto
Le chemin le plus court entre Anne et Alberto comporte deux étapes (Anne → Zoé et Zoé → Alberto). On dit donc qu'il y a 2 degrés de séparation entre eux. On peut aussi dire que Alberto est un ami d'une amie d'Anne.



Vous pouvez supposer qu'au départ, la figure précédente représente bien tous les liens d'amitié entre les personnes. Vous devrez placer ces liens dans votre programme. Or, ces liens peuvent changer et votre programme doit tenir compte de ces changements. Par exemple, des amitiés peuvent

commencer, possiblement avec de nouvelles personnes. Des amitiés peuvent se terminer. Votre programme doit être en mesure de trouver les amis des amis et de déterminer les degrés de séparation entre deux personnes.

Description des entrées et des sorties

Votre programme doit pouvoir lire six commandes possibles. Chaque commande indique une action particulière (voir ci-dessous). Vous pouvez supposer que x et y sont des entiers, que $x \neq y$, $x \geq 1$, $y \geq 1$, $x < 50$ et $y < 50$. Vous pouvez auss supposer que les commandes (`i`, `d`, `n`, `f`, `s`, `q`) sont présentées une par ligne et que les paramètres (zero, un ou deux nombres entier) paraissent une par ligne.

- `i` x y – Créer un lien d'amitié entre la personne x et la personne y. Si x et y sont déjà amis, ne rien ajouter. Si x ou y est une nouvelle personne, l'ajouter.
- `d` x y – Défaire le lien d'amitié entre la personne x et la personne y.
- `n` x – Sortir le nombre d'amis de la personne x.
- `f` x – Sortir le nombre d'« amis d'amis » de la personne x. Avis que x et les amis directes de x ne sont pas compter pour « amis d'amis ».
- `s` x y – Sortir le nombre de degrés de séparation entre x et y. S'il n'y a aucun chemin entre x et y, sortir `Aucun lien`.
- `q` – Quitter le programme.

Exemple d'une interaction

Entrée	Sortie	Explication

`i`
`20`
`10`
	(aucune sortie)	L'insertion d'un lien d'amitié ne produit aucune sortie.

`i`
`20`
`9`
	(aucune sortie)	L'insertion d'un lien d'amitié ne produit aucune sortie.

`n` `20`	`2`	La personne 20 a deux amis (10 et 9)
`f` `20`	`3`	Les amis des amis de 20 sont 8, 11, 12.

`s`
`20`
`6`
	`4`	Le plus court chemin est $20 \rightarrow 9 \rightarrow 8 \rightarrow 7 \rightarrow 6$.

`q`	(aucune sortie)	Le programme est terminé.