Formulaire d'inscription

À photocopier et à renvoyer à l'adresse postale suivante avant le 15 février 2012:

✉ Concours Informatique Luxembourgeois
p.a. Centre de Technologie de l'Éducation
29 avenue J.-F. Kennedy
L-1855 Luxembourg

Nom et prénom
Lieu et date de naissance
Adresse privée (rue et numéro, code postal et localité)

Adresse de courriel	Téléphone

ATTENTION: vérifiez bien votre adresse de courriel, car elle sera le seul moyen de communication entre les organisateurs du CIL et les candidats.

Etablissement scolaire	Classe

Langage(s) de programmation utilisé(s) - cochez ce qui convient: ☐ Pascal ☐ C ☐ C++

L'autorisation d'un des parents ou du représentant légal est indispensable pour l'inscription si le candidat est mineur.

Accord et signature d'un des parents ou du représentant légal (si le candidat est mineur).

Je soussigné(e) _____ m'inscris au Concours Informatique Luxembourgeois 2012 et m'engage à respecter le règlement ainsi que les conditions de participation.

Date et signature du candidat: _____

CENTRE DE TECHNOLOGIE DE L'ÉDUCATION *www.cte.lu*	**21e CONCOURS INFORMATIQUE LUXEMBOURGEOIS** *cil.cte.lu*	**24e OLYMPIADE INTERNATIONALE EN INFORMATIQUE** *www.ioi2012.org*

[Affiche: Pat Faust de la IIIe D du LCD]

Le Concours Informatique Luxembourgeois (CIL)

Le Concours Informatique Luxembourgeois (CIL) est un concours scolaire de programmation, organisé annuellement par le Centre de Technologie de l'Éducation (CTE). Le concours vise à identifier de jeunes élèves ayant des compétences particulières dans la résolution de problèmes sur ordinateur, dans la conception d'algorithmes et l'implémentation de programmes à l'aide d'un langage de programmation. En 2012, le CIL est organisé déjà pour la 21e fois. Le concours est ouvert à tous les élèves des lycées publics et privés des enseignements secondaire et secondaire technique, sans limite d'âge. Les langages de programmation admis lors du CIL sont le Pascal et le C/C++. Ces deux langages sont d'ailleurs aussi les langages de programmation officiels des Olympiades Internationales en Informatique (IOI). Le CIL 2012 est organisé en quatre étapes:

- l'Épreuve de Sélection Préliminaire,
- l'Épreuve Demi-Finale,
- un certain nombre de Séances de Formations et
- l'Épreuve Finale.

Toutes les informations relatives au concours peuvent être consultées sur le site web *cil.cte.lu*. Traditionnellement, les quatre lauréats du CIL font partie de la délégation luxembourgeoise officielle qui représente les couleurs du Grand-Duché de Luxembourg lors de l'Olympiade Internationale en Informatique (IOI).

L'Olympiade Internationale en Informatique (IOI)

L'Olympiade Internationale en Informatique (IOI) est une compétition internationale d'algorithmique et de programmation d'ordinateurs pour jeunes élèves. Elle est organisée chaque année dans un autre pays du monde et accueille actuellement les délégations de plus de 80 pays du monde entier. L'IOI est une des sept olympiades scientifiques internationales pour jeunes élèves (les autres concernent les mathématiques, la physique, la biologie, l'astronomie, la chimie et la géographie). Le Grand-Duché de Luxembourg participe déjà depuis 1992 aux IOI. Les langages de programmation officiels des IOI sont les mêmes que ceux du CIL. Les vainqueurs des IOI feront partie à l'avenir des meilleurs programmeurs au monde. La 24e édition de l'IOI se déroulera du 22 au 29 septembre à Sirmione en Italie. Pour plus d'informations sur l'IOI 2012, prière de consulter le site web *www.ioi2012.org*.

Calendrier

15 novembre 2011	Lancement officiel du 21e Concours Informatique Luxembourgeois (CIL) 2012
15 février 2012	Épreuve de Sélection Préliminaire: date limite des envois (formulaire d'inscription et solutions aux problèmes)
27 février 2012	Épreuve Demi-Finale
avril-juin 2012	Séances de Formation
5 juillet 2012	Épreuve Finale
juillet - septembre 2012	Formation algorithmique approfondie pour les quatre lauréats du CIL 2012
22-29 septembre 2012	24e Olympiade Internationale en Informatique (IOI) à Sirmione en Italie
septembre 2012	Remise des prix du CIL 2012

MINISTÈRE DE L'ÉDUCATION NATIONALE
ET DE LA FORMATION PROFESSIONNELLE
Centre de technologie de l'éducation

Déroulement du concours

La 21e édition du Concours Informatique Luxembourgeois (CIL) consiste en quatre étapes:

Étape I
Épreuve de Sélection Préliminaire

Les candidats doivent résoudre à domicile les problèmes énoncés dans le questionnaire ci-dessous. Le questionnaire se compose de quatre problèmes d'un degré de difficulté varié. Afin de pouvoir participer au concours, il faut avoir résolu au minimum deux problèmes (il n'est donc pas nécessaire de résoudre la totalité des problèmes posés). Les langages de programmation permis sont le Pascal (→ Turbo Pascal, Freepascal, fonctionnalité « console application » de Delphi etc.) ou le C/C++ (→ Turbo C/C++, GNU C/C++ etc.). Les solutions aux problèmes (les fichiers source et exécutable sont de rigueur pour chacune des solutions proposées) doivent parvenir aux organisateurs par courrier électronique jusqu'au 15 février 2012 au plus tard à l'adresse électronique suivante:

@ **Adresse électronique:**
cil@cte.lu

Afin d'éviter des problèmes de transmission, il est nécessaire d'envoyer les fichiers de manière compressée. Le formulaire d'inscription dument rempli et signé doit aussi parvenir aux organisateurs par courrier postal jusqu'au 15 février 2012 au plus tard. Pour les candidats mineurs, l'autorisation d'un des parents ou du représentant légal est indispensable pour l'inscription.

✉ **Adresse postale:**
Concours Informatique Luxembourgeois
p.a. Centre de Technologie de l'Éducation
29 avenue J.-F. Kennedy
L–1855 Luxembourg

Le questionnaire de l'Épreuve de Sélection Préliminaire ainsi que le formulaire d'inscription sont aussi disponibles sur le site web *cil.cte.lu*. Les candidats ayant réalisé les meilleurs scores à l'Épreuve de Sélection Préliminaire sont admis à l'Épreuve Demi-Finale.

Étape II
Épreuve Demi-Finale

L'Épreuve Demi-Finale consiste à résoudre individuellement trois problèmes d'un degré de difficulté varié dans un temps déterminé. Les candidats disposent de quatre heures pour concevoir les solutions aux problèmes d'algorithmique posés et pour implémenter les programmes correspondants en Pascal ou en C/C++. L'Épreuve Demi-Finale a lieu le 27 février 2012 dans une salle informatique d'un lycée. L'Épreuve Demi-Finale permet de sélectionner au plus douze candidats qui sont admis aux Séances de Formation.

Étape III
Séances de Formation

Les Séances de Formation permettent aux candidats sélectionnés d'approfondir leurs connaissances en programmation et de s'approprier des méthodes d'algorithmique. Les Séances de Formation, qui sont au nombre de quatre, se déroulent pendant les mois d'avril à juin 2012. En principe, tous les candidats ayant participé à au moins 75% des séances sont admis à l'Épreuve Finale.

Étape IV
Épreuve Finale

L'Épreuve Finale consiste à résoudre individuellement deux problèmes d'un degré de difficulté varié dans un temps déterminé. Les candidats disposent de quatre heures pour concevoir les solutions aux problèmes d'algorithmique posés et pour implémenter les programmes correspondants en Pascal ou en C/C++. Les problèmes posés requièrent en grande partie la mise en œuvre des méthodes d'algorithmique traitées lors des Séances de Formation. L'Épreuve Finale a lieu le 5 juillet 2012 dans uns salle informatique d'un lycée. Traditionnellement, l'Épreuve Finale permet de sélectionner les quatre lauréats du CIL qui, après une formation algorithmique appropriée, représenteront le Grand-Duché de Luxembourg à la 24e édition de l'Olympiade Internationale en Informatique (IOI) du 22 au 29 septembre 2012 à Sirmione en Italie.

Problème I — Porte-capsules

Un porte-capsules est un dispositif métallique qui permet de ranger aisément des capsules de café, comme on peut le voir sur la figure ci-contre. Votre tâche consiste dans la programmation d'une simulation d'un porte-capsules électronique.

Tâche

Écrivez un programme qui permet de simuler un porte-capsules de **N** colonnes numérotées de 1 à **N**, dont chaque colonne peut contenir au maximum **M** capsules. La couleur d'une capsule est représentée par une des 26 lettres majuscules de l'alphabet latin **C**. Deux capsules ont donc la même couleur lorsqu'elles sont représentées par la même lettre majuscule.

Dans la simulation les capsules sont soumises à la force de gravitation et glissent toujours vers le bas de leur colonne respective.

Votre simulation doit supporter les commandes suivantes:

- **PUT <colonne> <couleur>**

  Cette commande rajoute une capsule de la couleur <couleur> à la colonne numéro <colonne>. Une nouvelle capsule est toujours rajoutée en-haut de la colonne. La commande doit afficher le message d'erreur « Colonne pleine » si la colonne contient déjà le maximum de capsules. Exemples:

  - PUT 1 R — Rajoute une capsule « R » à la colonne numéro 1.
  - PUT 5 G — Rajoute une capsule « G » à la colonne numéro 5.

- **GET <colonne>**

  Cette commande enlève une capsule à la colonne numéro <colonne>. Une capsule est toujours enlevée en bas de la colonne. La commande doit afficher la couleur de la capsule enlevée ou le message d'erreur « Colonne vide » si la colonne est vide. Exemples:

  - GET 2 — Enlève une capsule de la colonne numéro 2 et affiche la lettre majuscule correspondant à sa couleur.
  - GET 6 — Enlève une capsule de la colonne numéro 6 et affiche la lettre majuscule correspondant à sa couleur.

- **FIND <couleur>**

  Cette commande recherche la capsule de la couleur <couleur> la plus facilement accessible par le bas dans tout le porte-capsules. La facilité d'accès d'une capsule est définie par le nombre de capsules qu'il faut enlever du bas d'une colonne pour atteindre cette capsule. La commande doit afficher le numéro de la colonne ainsi que le nombre de capsules qu'il faut enlever avant de pouvoir accéder à une capsule de la couleur recherchée ou le message d'erreur « Couleur non disponible » lorsqu'aucune capsule de la couleur <couleur> ne se trouve dans le porte-capsules. Lorsque plusieurs colonnes conviennent, la commande peut en choisir n'importe laquelle. Exemples:

  - FIND R — Affiche par exemple qu'il faut enlever 3 capsules de la colonne 5 avant de pouvoir accéder à une capsule « R ». Dans cet exemple, 3 doit être le minimum de toutes les colonnes du porte-capsules.
  - FIND Y — Affiche par exemple qu'il faut enlever 2 capsules de la colonne 1.

- **STOP**

  Cette commande termine la simulation et quitte l'application.

Restrictions

C = {A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z}

$1 \le \mathbf{N} \le 1\,000$ — Le nombre de colonnes dans le porte-capsules.

$1 \le \mathbf{M} \le 1\,000$ — La capacité en capsules d'une seule colonne.

Entrée et sortie du programme

- Le programme lit d'abord à partir du clavier le nombre de colonnes **N** du porte-capsules. Les **N** colonnes sont numérotées de 1 à **N**.
- Le programme lit ensuite la capacité en capsules **M** d'une seule colonne.
- Ensuite, la simulation du porte-capsules est interactive l'utilisateur peut entrer indéfiniment une des commandes PUT, GET, FIND ou la commande STOP et le programme doit réagir correctement à cette commande. L'utilisateur veille à respecter le bon format pour chaque commande les arguments des commandes PUT, GET, FIND sont séparés par une seule espace. Les noms des commandes sont toujours écrits en lettres majuscules.

Exemple d'exécution

```
Nombre de colonnes: 3
Capacité d'une colonne: 3
Commande: PUT 1 R
Commande: PUT 1 G
Commande: PUT 1 M
Commande: PUT 1 Y
La colonne 1 est pleine
```

```
Commande: PUT 2 G
Commande: GET 1
Vous avez enlevé une capsule 'R' de la colonne 1
Commande: PUT 3 M
Commande: FIND M
Il faut enlever 0 capsules de la colonne 3 pour accéder à
une capsule 'M'
Commande: FIND S
Le porte-capsules ne contient pas de capsule 'S'
Commande: GET 2
Vous avez enlevé une capsule 'G' de la colonne 2
Commande: GET 2
La colonne 2 ne contient pas de capsule
Commande: STOP
```

Remettez le programme sous le nom CAPSULES.xxx, avec xxx=PAS ou C(PP). Remettez également le fichier binaire exécutable CAPSULES.EXE correspondant au programme.

Problème II — Synthèse vocale

La synthèse vocale est une technique informatique de synthèse sonore qui permet de créer de la parole artificielle à partir de n'importe quel texte. Elle est utilisée notamment dans les assistants de navigation personnels comme les GPS (Global Positioning System) dans les automobiles. Cependant, avant de pouvoir synthétiser des nombres, il faut d'abord convertir ces nombres en un texte prononçable.

Au lieu de prononcer les nombres comme par exemple

un GPS le prononce correctement comme par exemple

Tâche

Écrivez un programme qui permet de « prononcer » correctement en allemand n'importe quel nombre entier **X**. Votre programme doit convertir le nombre entier **X** en un texte qui, lu à haute voix, correspond exactement à la prononciation de ce nombre.

Restrictions

$1 \leq X \leq 10^{24}$ La valeur du nombre entier **X** à prononcer.

Entrée et sortie du programme

Entrée

- Le programme lit à partir du clavier un seul nombre entier **X**, qui représente le nombre à convertir en une version textuelle.

Sortie

- Le programme affiche à l'écran une seule ligne qui représente la version textuelle du nombre **X**. Référez-vous au lien suivant pour trouver le nom des nombres:

de.wikipedia.org/wiki/Zahlennamen

Exemples d'exécution

Exemple 1

```
111
einhundertelf
```

Exemple 2

```
901000000
neunhunderteins Millionen
```

Exemple 3

```
1000
eintausend
```

Exemple 4

```
627
sechshundertsiebenundzwanzig
```

Exemple 5

```
101213501
einhunderteins Millionen zweihundertdreizehn Tausend fünf-
hunderteins
```

Remettez le programme sous le nom SYNTHESE.xxx, avec xxx=PAS ou C(PP). Remettez également le fichier binaire exécutable SYNTHESE.EXE correspondant au programme.

Problème III — ISBN

L'ISBN (International Standard Book Number) ou numéro international normalisé du livre est un numéro international qui permet d'identifier, de manière unique, certains livres publiés. Il est destiné à simplifier la gestion informatique du livre bibliothèques, libraires, distributeurs, etc.

Une version particulière du numéro ISBN est la norme ISBN-10, qui définit que les numéros doivent être formés d'exactement dix chiffres. Les neuf premiers chiffres sont compris entre 0 à 9 (bornes incluses). Le dernier caractère est un code clé de vérification calculé à partir des chiffres précédents. Outre les chiffres de 0 à 9, le code clé de vérification peut prendre la valeur X, qui représente le nombre 10.

Tâche

Écrivez un programme qui permet de reconstituer un numéro ISBN-10 dont il manque un caractère. Le caractère manquant est indiqué par un '?' (point d'interrogation).

Afin de vérifier la validité d'un numéro ISBN-10, on attribue une pondération à chaque position (de 10 à 1 en allant en sens décroissant) et on fait la somme des produits ainsi obtenus. Pour être un numéro ISBN-10 valide, cette somme doit être un multiple de 11.

Exemple pour l'ISBN-10 « 2940199612 » de la figure ci-dessus, on vérifie que

$$2\cdot10 + 9\cdot9 + 4\cdot8 + 0\cdot7 + 1\cdot6 + 9\cdot5 + 9\cdot4 + 6\cdot3 + 1\cdot2 + 2\cdot1 = 242 = 22\cdot11$$

Ce numéro ISBN-10 est donc bien correct.

Restrictions

- Les neuf premiers caractères sont des chiffres compris entre 0 et 9 (bornes incluses).
- Le dixième caractère est un chiffre compris entre 0 et 9 (bornes incluses) ou le caractère 'X' représentant la valeur numérique 10.
- Le caractère manquant est représenté par le caractère '?'. Le nombre de caractères manquants est fixé à 1. Votre programme ne doit pas vérifier le nombre de caractères manquants.

Entrée et sortie du programme

Entrée

- Le programme lit à partir du clavier un numéro ISBN-10 composé exactement de 10 caractères sans espaces, ni tirets, etc.

Sortie

- Le programme affiche à l'écran le numéro ISBN-10 reconstitué, c.-à-d. le numéro ISBN-10 de l'entrée dans lequel le caractère manquant a été remplacé par le bon caractère.

Exemples d'exécution

Exemple 1

```
3746237?32
3746237432
```

Exemple 2

```
02?1722313
0201722313
```

Exemple 3

```
020170353?
020170353X
```

Exemple 4

```
02017023?X
020170238X
```

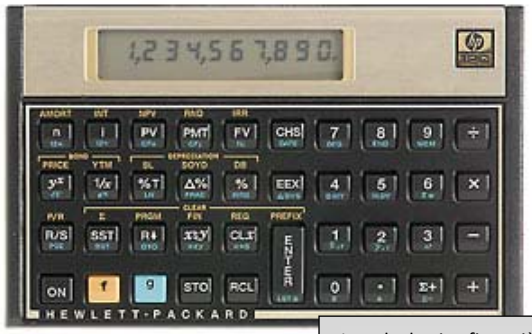
Remettez le programme sous le nom ISBN.xxx, avec xxx=PAS ou C(PP). Remettez également le fichier binaire exécutable ISBN.EXE correspondant au programme.

Problème IV — Infix2Postfix

Les notations infixée (ou infixe), préfixée (ou préfixe) et postfixée (ou postfixe) sont des formes d'écritures d'expressions algébriques qui se distinguent par la position relative qu'y prennent les opérateurs et leurs opérandes. Un opérateur est écrit avant ses opérandes en notation préfixée, entre ses opérandes en notation infixée et après ses opérandes en notation postfixée.

La notation infixée n'a de sens que pour les opérateurs prenant exactement deux opérandes. C'est la notation la plus courante de tels opérateurs en mathématiques.

La notation préfixée fut inventée en 1920 par le mathématicien polonais Jan Łukasiewicz, c'est pourquoi elle est également appelée notation polonaise, et la notation postfixée, par opposition, notation polonaise inverse. Ces deux notations permettent de ne pas utiliser de parenthèses. Un autre avantage de la notation polonaise inverse est qu'il faut taper moins de touches pour calculer une expression.



La calculatrice financière HP-12C produite depuis 1981 jusqu'à aujourd'hui utilise la notation polonaise inverse.

Tâche

Écrivez un programme qui permet de convertir une expression en notation infixée correctement formée en une expression en notation postfixée équivalente.

Exemples

Notation infixée	Notation postfixée ou polonaise inverse
(1+2)*(3+4)	1 2 + 3 4 + *
5 + ((1 + 2) * 4) − 3	5 1 2 + 4 * + 3 −

Restrictions

- Il faut uniquement tenir compte des opérateurs + − * / (addition, soustraction, multiplication et division).
- Les opérandes sont des entiers positifs.
- Les opérateurs * et / sont prioritaires par rapport à + et −. Les opérateurs de même priorité sont évalués de la gauche vers la droite. L'utilisation de paires de parenthèses permet de modifier l'ordre d'évaluation.

Entrée et sortie du programme

Entrée

- Le programme lit à partir du clavier une expression correctement formée en notation infixée. Cette expression ne contient aucun séparateur (espaces, tabulation, etc.) et ne dépasse pas 255 caractères.

Sortie

- Le programme affiche à l'écran l'expression entrée, convertie en notation postfixée. Lors de l'affichage respectivement d'un opérande et d'un opérateur celui-ci est toujours suivi par un caractère ' ' (espace).

Exemples d'exécution

Exemple 1

```
12/3
12 3 /
```

Exemple 2

```
(1+2)*(3+4)
1 2 + 3 4 + *
```

Exemple 3

```
51-(725/42)
51 725 42 / -
```

Exemple 4

```
5+((1+2)*4)-3
5 1 2 + 4 * + 3 -
```

Remettez le programme sous le nom INFIX2POSTFIX.xxx, avec xxx=PAS ou C(PP). Remettez également le fichier binaire exécutable INFIX2POSTFIX.EXE correspondant au programme.