Concours Informatique Luxembourgeois 2005
Epreuve de Sélection Préliminaire

Cotation sur 100 points

VERSION 1.01

Problème I – Racine n-ième 15

Ecrivez un programme qui calcule la racine n-ième d'un nombre décimal.

Définition

$\sqrt[n]{x} = x^{\frac{1}{n}}$ avec: $n \in IN^*$, $n \geq 2$, appelé l'indice
$x \in ID$, appelé le radicant

Exemples

$\sqrt[2]{169} = 13$

$\sqrt[4]{-16}$ n'existe pas [indice pair, radicant négatif]

$\sqrt[3]{-125} = -5$ [indice impair, radicant négatif]

$\sqrt[8]{256} = 2$

$\sqrt[9]{-512} = -2$

$\sqrt[4]{16} = 2$

Restrictions

On suppose que les deux restrictions suivantes sont toujours vraies. Votre programme n'a donc pas besoin d'effectuer des tests de débordement.

- $2 \leq n \leq 100$
- $-1000000 \leq x \leq 1000000$

Entrées et sorties du programme

⇨ Le programme lit le radicant (suivi de ↵) et l'indice (suivi de ↵) du clavier.

⇦ Le programme écrit le résultat sur l'écran. En cas d'opération impossible, le résultat est le texte "CETTE RACINE N'EXISTE PAS!".

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

Problème II – Produit cartésien 20 points

Ecrivez un programme qui calcule le produit cartésien des éléments de deux listes numériques.

Exemple

Liste A: | 12 | 1 | 32 | -9 | 0 | 15 | 6 | -4 | 100 | 12 | 4 |

Liste B: | 3 | 87 | 6 | -25 | 127 | 56 | 0 | 8 | 8 | 61 | -1 |

Le produit cartésien P des éléments de deux listes est la somme des produits des éléments.
P = 12×3 + 1×87 + 32×6 + (-9)×(-25) + 0×127 + 15×56 + 6×0 + (-4)×8 + 100×8 + 12×61 + 4×(-1)
....

Restrictions

On suppose que les trois restrictions suivantes sont toujours vraies. Votre programme n'a donc pas besoin d'effectuer des tests de débordement.

- Les éléments des deux listes sont des nombres entiers de l'intervalle [-128;127].
- La longueur des deux listes, c.-à-d. le nombre d'éléments, est comprise entre 1 et 255.
- Les deux listes ont la même longueur.

Entrées et sorties du programme

⇨ Le programme lit les éléments de la liste A d'un fichier texte LISTEA.TXT et les éléments de la liste B d'un fichier texte LISTEB.TXT.

Dans les deux fichiers, les éléments sont séparés par un espace.

⇦ Le programme écrit le résultat sur l'écran. En cas d'opération impossible, le résultat est le texte "CETTE RACINE N'EXISTE PAS!".

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

Problème II – Langage LISP 25 points

Il s'agit de calculer les valeurs d'expressions arithmétiques d'un sous-ensemble du langage de programmation LISP, qui est utilisé dans le domaine de l'intelligence artificielle. Les expressions sont formées par des fonctions qui possèdent toujours 2 arguments, peuvent être imbriquées et doivent être parenthésées correctement.

Exemples

(ADD 10 8)	donne 18
(ADD 14(ADD -28 5))	donne -9
(MUL(ADD 2 -8)12)	donne -72
(DIV(ADD 3 12)(SUB 6 1))	donne 3
(MUL 18(DIV(MUL(SUB 15 9)2)3))	donne 4

Syntaxe des fonctions

(ADD <arg1> <arg2>)	avec <arg1> et <arg2> des constantes entières ou des résultats entiers d'une autre fonction
	retourne la somme des valeurs de <arg1> et de <arg2>
(MUL <arg1> <arg2>)	avec <arg1> et <arg2> des constantes entières ou des valeurs entières d'une autre fonction
	retourne le produit des valeurs de <arg1> et de <arg2>
(SUB <arg1> <arg2>)	avec <arg1> et <arg2> des constantes entières ou des valeurs entières d'une autre fonction
	retourne la différence des valeurs de <arg1> et de <arg2>
(DIV <arg1> <arg2>)	avec <arg1> et <arg2> des constantes entières ou des valeurs entières d'une autre fonction (<arg2>≠0)
	retourne le quotient entier des valeurs de <arg1> et de <arg2>

Restrictions

On suppose que les cinq restrictions suivantes sont toujours vraies.

- Les expressions données sont toujours syntaxiquement correctes. Votre programme n'a donc pas besoin de tester si le nombre des argments est correct, si les parenthèses sont bien positionnées etc.
- Les constantes entières sont dans l'intervalle [-32786; 32767]. Votre programme n'a donc pas besoin d'effectuer des tests de débordement.
- Les résultats des fonctions tombent toujours dans cet intervalle. Votre programme n'a donc pas besoin d'effectuer des tests de débordement.
- Dans les expressions, il n'y a pas d'espace - sauf entre deux constantes consécutives ainsi qu'entre le nom d'une fonction et une constante.
- La longueur maximale d'une expression est de 255 caractères (espaces inclus).

Entrées et sorties du programme

⇨ Le programme lit du fichier texte LISP_IN.TXT l'expression (sur une seule ligne).

⇦ Le programme écrit le résultat dans le fichier texte LISP_OUT.TXT. En cas de division par zéro, le résultat est le texte "ERREUR - DIVISION PAR ZERO!".

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

Problème IV – Echecs 40 points

Soit un échiquier avec 8×8 cases et les pièces suivantes:

- au maximum 2 tours (T), qui peuvent se déplacer d'un nombre quelconque de cases libres sur les lignes et les colonnes,
- au maximum 2 fous (F), qui peuvent se déplacer d'un nombre quelconque de cases libres sur les diagonales,
- la dame (D), qui peut se déplacer d'un nombre quelconque de cases libres dans toutes les directions (lignes, colonnes et diagonales).

Le roi (R) est en prise (en échec) s'il se trouve dans la ligne de déplacement possible d'au moins une des pièces ci-dessus.

Exemple I

	A	B	C	D	E	F	G	H
1								
2		D						
3						F		
4							T	
5								
6				R				
7	F							
8								

Le roi n'est pas en échec.

Exemple II

	A	B	C	D	E	F	G	H
1			R					
2					F			T
3								
4								
5								
6								F
7								
8		T						

Le roi est en échec, provoqué par le fou en H6.

Exemple III

	A	B	C	D	E	F	G	H
1		T						
2								
3					D			
4							F	
5			F					
6		R						
7								
8								

Le roi est en échec, provoqué par la tour en B1 et par le fou en C5.

Attention: on ne considère ni la couleur des cases, ni celle des pièces. En plus, on n'a pas besoin d'analyser si les situations de jeu sont réelles ou non.

Ecrivez un programme qui détermine combien de fois le roi est en échec, suite à une certaine position des autres pièces sur l'échiquier.

Entrées et sorties du programme

⇨ Le programme lit du fichier texte ECHEC_IN.TXT la position des pièces:

<ligne du roi><colonne du roi> <ligne de la 1ère tour><colonne de la 1ère tour> <ligne de la 2e tour><colonne de la 2e tour> <ligne du 1er fou><colonne du 1er fou> <ligne du 2e fou><colonne du 2e fou>

En cas d'absence d'une dame, d'une tour ou d'un fou, les valeurs pour la colonne et la ligne se limitent à "00".

⇦ Le programme écrit le résultat dans le fichier texte ECHEC_OUT.TXT. Si le roi n'est pas en échec, le résultat est 0, sinon le résultat correspond au nombre de fois que le roi est en échec.

Exemples d'entrées et de sorties

Pour l'exemple I ci-dessus, le fichier d'entrée est:

```
D6 B2 G4 00 F4 A7
```

Le fichier de sortie est:

```
0
```

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