Bistromathique

Version du 9 November 2002

Assistants C/Unix

> The Bistromatic Drive is a wonderful new method of crossing vast interstellar distances without all that dangerous mucking about with Improbability Factors.
>
> Bistromathics itself is simply a revolutionary new way of understanding the behaviour of numbers. Just as Einstein observed that time was not an absolute but depended on the observer's movement in space, and that space was not an absolute, but depended on the observer's movement in time, so it is now realized that numbers are not absolute, but depend on the observer's movement in restaurants.
>
> The first non-absolute number is the number of people for whom the table is reserved. This will vary during the course of the first three telephone calls to the restaurant, and then bear no apparent relation to the number of people who actually turn up, or to the number of people who subsequently join them after the show/match/party/gig, or to the number of people who leave when they see who else has turned up.
>
> The second non-absolute number is the given time of arrival, which is now known to be one of those most bizarre of mathematical concepts, a recipriversexcluson, a number whose existence can only be defined as being anything other than itself. In other words, the given time of arrival is the one moment of time at which it is impossible that any member of the party will arrive. Recipriversexclusons now play a vital part in many branches of maths, including statistics and accountancy and also form the basic equations used to engineer the Somebody Else's Problem field.
>
> The third and most mysterious piece of non-absoluteness of all lies in the relationship between the number of items on the bill, the cost of each item, the number of people at the table, and what they are each prepared to pay for. (The number of people who have actually brought any money is only a sub-phenomenon in this field.)

(extrait du *Hitch-Hiker's Guide to the Galaxy*, par feu Douglas Adams)

Il s'agit de réaliser une machine qui évalue une expression arithmétique où interviennent des nombres arbitrairement grands, dans une base quelconque.

1 Considérations génerales

Les sources de votre projet doivent être conformes au CSS, tant au niveau de la mise en forme que de la structure du répertoire de rendu (‘`Makefile`’, ‘`AUTHORS`’, etc...)

Votre programme devra compiler *et* fonctionner sur *toutes* les architectures de l’école.

Vous ne devez rien afficher de plus ou de moins que ce qui est demandé dans le sujet. Vous serez corrigés par une machine, qui ne tolèrera aucun écart.

1.1 Fonctions autorisées

Vous n’avez droit qu’aux fonctions suivantes :

`malloc(3)` `realloc(3)` `free(3)` `read(2)` `write(2)` `exit(3)` `assert(3)`

1.2 Modalités du projet

Ce projet est à réaliser en binôme. Ces binômes doivent être formés au plus tard une semaine après l’annonce du sujet. Vous devez annoncer la composition de votre binôme en envoyant un mail à `acu@epita.fr`, mail dont le titre doit respecter le format suivant :

```
[BISTRO] login1:login2
```

Vous devez stocker vos sources dans le répertoire ‘`~/c/rendu/proj/bistromathique`’, en les protégeant contre l’accès par d’autres utilisateurs[1].

Les règles de votre fichier ‘`Makefile`’ doivent générer un exécutable nommé ‘`bistromathique`’.

Vous devez rendre votre projet sous forme d’archive cryptée, comme l’indique la page correspondante sur le site des ACUs, avant le *lundi 11 novembre à minuit*.

1.3 Critères d’évaluation

Ceci est un sujet à plusieurs étages. La gestion de chaque étage doit être parfaite pour que le suivant soit pris en compte.

Les performances de la bistromathique relativement à la machine de référence déterminent la note finale.

1.3.1 Détail du mécanisme d’évaluation

Votre programme sera évalué comme suit :

- à la moindre erreur de conformité de vos sources avec le CSS ou les modalités de rendu, vous vous verrez attribuer la note définitive et non contestable de 0 ;
- si vous ne rendez rien, ou s’il y a des erreurs de compilation, vous vous verrez attribuer la note définitive et non contestable de -5, pour “fainéantise” ;
- si tout se passe bien, votre bistromathique aura droit à une notation à étages :
  1. l’addition et la soustraction *entières* seront exhaustivement testées, toute erreur à cette étape étant éliminatoire[2] ;
  2. votre multiplication *entière* sera testée et ses performances comparées avec la plus rapide des bistromathiques[3]. Si elle fonctionne bien, une quantité de points corrélée[4] à sa rapidité relative sera attribuée ;

[1] droits 600 et 700

[2] 0 en note finale

[3] c’est-à-dire la machine de référence, ou la plus rapide machine de votre promotion si elle est plus rapide que la machine de référence

[4] attention, la corrélation n’est pas linéaire

3. votre division *entière* sera testée et ses performances comparées avec la plus rapide des bistromathiques. Si elle fonctionne bien, une quantité de points corrélée à sa rapidité relative sera attribuée ;
4. si tout fonctionne jusque là, nous procéderons aux deux étapes précédentes en arithmétique fractionnelle *sans* obligation de fournir le résultat sous forme irréductible. Le bon fonctionnement de ces opérations sera récompensé ;
5. si tout fonctionne jusque là, nous procéderons aux deux étapes, *avec* l'obligation de fournir les résultats sous forme irréductible, et en tenant compte des performances de la machine.

Tous ces tests sont effectués en notation RPN**.**

- une batterie de tests séparés éprouvera, si elle existe, la bonne gestion par votre bistromathique de la notation infixe. Si la notation infixe est gérée correctement, la note finale est multipliée par 1.25 ;
- si vous essayez de détourner le sujet, ou de produire un résultat sans que votre code C n'en soit responsable, vous vous verrez attribuer la note de -21, pour "foutage de gueule".
- la tricherie (utilisation du travail d'autrui en le faisant passer pour vôtre) est sanctionnée de la note -42 et d'un entretien avec les responsables pédagogiques.

Le détail des points attribués à chaque test et les corrélations entre performances et récompenses seront communiquées sur les news ; cependant, une bistromathique qui gère *parfaitement* les opérations arithmétiques *entières* dans la notation RPN (même lentement) pourra prétendre se voir récompenser d'au moins 10/20.

1.4 Documentation complémentaire

Vous disposez d'un newsgroup dedié au projet : '`epita.cours.c-unix.bistromatique`'[5]

Une FAQ sera mise en place, nous vous indiquerons dans le newsgroup ad-hoc son emplacement lors de sa création.

[5] oui, il y a une faute dans le nom du newsgroup, mais il est comme ca sur le serveur de news, on ne peut rien y changer, désolés pour la confusion éventuelle...

2 Expressions

Les expressions que votre bistromathique doit évaluer peuvent être données sous deux formes. La gestion de la deuxième forme est un *bonus* par rapport à la première, qui permet de gagner plus de points, mais qui ne sera considéré que si la gestion de la première forme est parfaite.

2.1 Forme de base - R.P.N.

Les expressions sont données dans la notation dite "polonaise inversée"[1], où les opérations que subissent les nombres sont indiquées *après* leurs opérandes.

Voici un exemple d'expression en RPN :

```
234233 2324244432 234243243 234234243 23423423+/+*
```

Dans cette notation, 7 opérateurs différents peuvent être utilisés : l'empilage[2], l'échange d'ordre, la négation, l'addition, la soustraction, la multiplication, la division et la division entière.

2.1.1 Exemples

Voici quelques exemples d'expressions et leurs résultats, dans la base décimale connue, et en utilisant les opérateurs ' ', '.', 'N', '+', '-', '*', '/', '\' :

```
10 5+3/
⇒ 5

10N7+3/
⇒ 1N

10 7+3/
⇒ 17 3/

10 7+3\
⇒ 5

10 7+3/1\
⇒ 5

1 0.-
⇒ 1N

42 3\4+69.-
⇒ 51
```

2.2 Forme étendue - infixe

Les expressions sont données dans la forme connue à opérateurs infixes, placés *entre* leurs opérandes, en utilisant les priorités du langage C.

Voici un exemple, équivalent à celui de la section précédente :

```
234233*(2324244432+234243243/(234234243+23423423))
```

Dans cette notation, 5 opérateurs peuvent être utilisés : l'addition, la négation/soustraction[3], la multiplication, la division et la division entière.

[1] RPN est l'acronyme de *Reverse Polish Notation*

[2] qui permet surtout de séparer deux nombres

[3] un seul opérateur pour les deux usages

Par ailleurs, les expressions peuvent être groupées (“parenthésées”) par deux opérateurs consacrés : le début et la fin de groupement.

2.2.1 Exemples

Voici quelques exemples d'expressions et leurs résultats, dans la base décimale connue, et en utilisant les opérateurs ‘`+`’, ‘`-`’, ‘`*`’, ‘`/`’, ‘`\`’, ‘`(`’ et ‘`)`’ :

```
(10+5)/3
⇒ 5

(-10+7)/3
⇒ -1

(10+7)/3
⇒ 17/3

(10+7)\3
⇒ 5

(10+7)/3\1
⇒ 5

0-1
⇒ -1

69-(42\3+4)
⇒ 51
```

Vous remarquerez l'équivalence de ces exemples avec ceux de la section précédente.

2.3 Calculs

Tous les calculs doivent être effectués en arithmétique *rationnelle*. Le résultat doit bien entendu être donné sous forme irréductible[4] (see Section 3.2 [Format de sortie], page 6).

Cependant, pour rendre complet l'exercice, il vous est demandé d'implémenter la division entière, qui réalise en arithmétique entière la division de ses deux opérandes.

À ce propos, remarquez bien que si $a = p/q, b = p'/q'$, alors $a\backslash b = (p * q')\backslash(q * p')$.

Dans la notation infixe, les deux opérateurs de division ont la *même* priorité.

Vous ne devez pas implémenter l'opérateur “modulo”.

[4] modulo le travail que vous êtes prêt à fournir (see Section 1.3 [Criteres d evaluation], page 2)

3 Entrées-sorties

Votre bistromathique doit pouvoir travailler avec n'importe quelle expression arithmétique, dans n'importe quelle base, et avec n'importe quelle représentation des opérateurs.

Les arguments vous seront donnés sur l'entrée standard, tandis que le résultat[1] devra être affiché sur la sortie standard.

3.1 Format d'entrée

Les arguments vous sont passés dans l'ordre et la forme suivants :

1. un octet qui vaut 51 si l'entrée est sous la forme R.P.N., ou 69 si l'entrée est sous la forme infixe ;
2.
   - si l'entrée est sous forme R.P.N., les opérateurs, dans l'ordre suivant : empilage, échange, négation, addition, soustraction, multiplication, division puis division entière ;
   - si l'entrée est sous forme infixe :
     - les deux opérateurs de groupement, début puis fin ;
     - les opérateurs arithmétiques, dans l'ordre suivant : addition, soustraction/négation, multiplication, division puis division entière ;
3. un octet spécifiant la taille de la base ;
4. la base ;
5. l'expression d'entrée.

3.1.1 Remarques importantes

- les différents champs de l'entrée ne sont pas séparés par un délimiteur particulier, et le caractère de retour la ligne et le caractère nul deviennent donc des caractères comme les autres ;
- pour la base et les opérateurs, aucun caractère n'est interdit.

3.1.2 Garanties

Un caractère donné ne figurera pas à la fois dans la base et comme opérateur.

Chaque caractère de la base sera unique dans la base.

Un caractère donné ne peut pas représenter deux opérateurs dans une même entrée.

3.2 Format de sortie

Le résultat ou le message d'erreur doivent être affichés sur la sortie standard, sans ajouter de retour la ligne terminal.

3.2.1 Forme R.P.N.

Si l'expression d'entrée est donnée sous la forme RPN, l'affichage du résultat doit se conformer aux contraintes suivantes :

- si le résultat a un dénominateur non unitaire, le nominateur et le dénominateur seront affichés l'un après l'autre. Le nominateur sera suivi de l'opérateur d'empilage, et le dénominateur de l'opérateur de division ;
- si le dénominateur du résultat vaut 1, seul le nominateur sera affiché, et ne sera pas suivi de l'opérateur d'empilage ;
- si le résultat est négatif, l'opérateur de négation sera affiché en fin de résultat.

[1] ou le message d'erreur...

3.2.2 Forme infixe

Si l'expression d'entrée est donnée sous la forme infixe, l'affichage du résultat doit se conformer aux contraintes suivantes :

- si le résultat a un dénominateur non unitaire, le nominateur et le dénominateur seront affichés l'un après l'autre, *séparés* par l'opérateur de division ;
- si le dénominateur du résultat vaut 1, seul le nominateur sera affiché ;
- si le résultat est négatif, l'opérateur de négation/soustraction sera affiché en tête du résultat.

4 Gestion des erreurs

Bien évidemment, la machine est capable de gérer les erreurs comme toute calculatrice de notre époque qui se respecte.

4.1 Mécanisme d'erreur

Quand une erreur survient, le message correspondant doit s'afficher sur la sortie standard en lieu et place du résultat, et le programme doit se terminer avec un code d'erreur non nul.

Il faut afficher un message différent par type d'erreur.

4.2 Exemples de condition d'erreur

4.2.1 Notation R.P.N.

- Dans la base '`artye`', '`rya a /`' provoque une *erreur arithmétique*, car '`a`' est l'élement neutre de l'addition dans la base ;
- Dans la même base, l'expression isolée '`a a + +`' provoque une *erreur de pile* ;
- De mme pour l'expression isolée '`a a a +`'.

4.2.2 Notation infixe

- Dans la base '`artye`', '`rya/a`' provoque une *erreur arithmétique* ;
- L'expression '`a*/b`' provoque une *erreur de syntaxe* ;
- Si la bistromathique ne gère pas la notation infixe, une entrée sous forme infixe provoque une *erreur de gestion.*

4.3 Les messages d'erreur

Il existe plusieurs messages d'erreur, chacun correspondant à une condition d'erreur différente.

`E_STACK` le message d'erreur de pile ;

`E_I_AM_WEAK`
: le message d'erreur de gestion ;

`E_ARITH` le message d'erreur arithmétique ;

`E_SYNTAX` le message d'erreur de syntaxe ;

`E_THE_WORLD_IS_OVER`
: le message d'erreur "système" (allocations, entrées-sorties ...).

4.4 Paramétrage des messages d'erreur

Le texte des messages d'erreur vous seront passés sous la forme de constantes chaîne définie dans un en-tête de déclaration nommé '`errors.h`', qui sera introduit de force dans votre répertoire de rendu lors de la correction.

Voici un exemple pour ce fichier :

```
#ifndef ERRORS_H
# define ERRORS_H

# define E_STACK                "Oops."
# define E_I_AM_WEAK            "Beat me hard."
# define E_ARITH                "You fool."
# define E_SYNTAX               "Duh ?"
# define E_THE_WORLD_IS_OVER    "I feel sick."

#endif
```

Index et Table des matières

Table of Contents