Projeto I - Avaliador de Expressões

O Projeto I é um **avaliador de expressões**. Ele deverá avaliar as entradas dadas e constatar se são ou não expressões bem formadas, em caso negativo a string "Expressao mal-formada" deverá aparecer na saída, em caso positivo a string "Expressao bem-formada" deverá aparecer na saída. Além disso, o avaliador deve informar ainda a altura da árvore sintática, o número de sub-expressões e informar para que valores das variáveis x, y, z, a expressão é VERDADEIRA. A lista de valores das variáveis precisa seguir uma ordem pré-definida especificada abaixo.

O avaliador poderá ser feito em qualquer linguagem de sua escolha.

Você deverá entregar somente o código fonte do seu avaliador.

O avaliador deve ser entregue **compilando**, portanto, antes de enviar teste-o nas máquinas do CIn.

O Avaliador de expressões deve ser entregue zipado por email para o seu monitor, com o assunto:
[Avaliador] - *login* (onde login corresponde ao login do aluno no CIn).

Este exercício é individual. Qualquer tentativa de fraude ou cópia será punida com uma nota 0 (ZERO) para ambos os infratores.

O prazo de entrega do avaliador é ATÉ as 23h59 do dia 05/03/2013. A data de entrega não será prorrogada.

O Avaliador será apresentado ao respectivo monitor, numa data ainda a confirmar.

A nota será da seguinte forma: **50% Implementação + 50% Apresentação**.

Detalhes

■ Quanto aos operadores:

Não há operadores fixos. Os operadores serão definidos no arquivo de entrada. Ou seja, podem surgir operadores com tabelas-verdade diferentes das existentes na lógica clássica.

Os operadores não terão o mesmo nome de uma variável ou constante, já definidos aqui.

■ Quanto ao número de sub-expressões:

O avaliador deve contar o número de sub-expressões não repetidas no conjunto de sub-expressões.
Assim, se uma expressão possuir (X.X), não se deve contar a sub-expressão "X" duas vezes.
A própria expressão entra no conjunto de sub-expressões.

■ **Quanto às variáveis:**

Só serão utilizadas 3 (três) variáveis:

x	y	z

■ **Quanto às constantes:**

TRUE	1
FALSE	0

■ **Quanto aos parênteses:**

TODO operador deve ser separado por parênteses.

Operador Unário	Operador Binário
(-x)	(x.y)
(-(x+y))	(x+("sub-expressão"))
(-("sub-expressão"))	("sub1"+"sub2")

■ **Quanto à Entrada/Saída**

Nome do arquivo: "Avaliador.xxx"

Entrada**:** Nome do arquivo "**Expressoes.in**"

Contem várias expressões que serão avaliadas.
A 1ª linha do arquivo de entrada conterá o número de operadores que serão passados para serem avaliados. Numa linha abaixo, será passado o operador, seguido por sua aridade, e na linha abaixo a isto, será passado a tabela verdade daquele operador. Ao final de todos os operadores e suas tabelas verdades, será passado o número de expressões que contém no arquivo de entrada e, ao pular a linha, as expressões a serem avaliadas estarão escritas.

Saída: Nome do arquivo "**Expressoes.out**"

Deve possuir TODAS as saídas do avaliador em um único arquivo, separadas por uma linha em branco.

Exemplos

Exemplo 0:

Arquivo: “Expressoes.in” (o que está depois das barras não estará no arquivo)

```
1   //número de operadores
- 1   //operador “-“ com aridade 1
1 0
0 1       //tabela verdade
1  //número de expressões a serem avaliadas
(-x)  //expressão a ser avaliada
```

Exemplo 1 ----------

Arquivo: "Expressoes.in"

```
1
- 1
1 0
0 1
1
(-x)
```

Arquivo: "Expressoes.out"

```
Expressao 1
Expressao bem-formada
Altura=1
Sub-expressoes=2
(X=0)
{linha em branco}
```

Exemplo 2 ----------

Arquivo: "Expressoes.in"

```
2
- 1
1 0
0 1
. 2
```

```
0 0  0
1 0 0
0 1 0
1 1 1
2
(-(x.(-z)))
-x
```

Arquivo: "Expressoes.out"

```
Expressao 1
Expressao bem-formada
Altura=3
Sub-expressoes=5
(X=0, Z=0); (X=0, Z=1); (X=1, Z=1)
{linha em branco}
Expressao 2
Expressao mal-formada
{linha em branco}
```

Exemplo 3 ----------

Arquivo: "Expressoes.in"

```
1
> 2
0 0 1
0 1 1
1 0 0
1 1 1
1
((x+y)+(0.1()
```

Arquivo: "Expressoes.out"

```
Expressao 1
Expressao mal-formada
{linha em branco}
```

Exemplo 4 ----------

Arquivo: "Expressoes.in"

```
3
+ 2
0 0 0
0 1 1
1 0 1
1 1 1
- 1
0 1
1 0
. 2
0 0 0
0 1 0
1 0 0
1 1 1
1
((x.x)+(-z))
```

Arquivo: "Expressoes.out"

```
Expressao 1
Expressao bem-formada
Altura=2
Sub-expressoes=5
(X=0, Z=0); (X=1, Z=0); (X=1, Z=1)
{linha em branco}
```

Dúvidas Frequentes

■ Não podem ser usadas funções prontas (como pacotes em Java ou bibliotecas em C/C++) para avaliar ou facilitar a avaliação de expressões.

■ A saída deve ser exatamente como está demonstrada na especificação.

■ As expressões de entrada terão um tamanho máximo de 100 caracteres.

■ Não haverá nenhum acento nem ponto na saída.

■ Se a expressão for insatisfatível ou for formada apenas por constantes (sendo tautologia ou insatisfatível), escreva uma linha em branco no lugar da valoração.

■ Não haverá espaços no meio da expressão.

■ As valorações devem seguir a ordem da tabela, ou seja: 000, 001, 010, 011... etc. Caso não haja uma constante, na valoração, o mesmo se repete: 00, 01, 10, 11. (veja exemplo quatro)

■ O arquivo de entrada não possui limite de expressões. Logo, teste seu programa com o máximo de expressões que puder.

Boa sorte a todos.

Qualquer dúvida mandem email para alunos_logica_ec@googlegroups.com com assunto: **[Especificação Avaliador]**. Ou procurem um monitor (e-mail ou pessoalmente).