MC202 — ESTRUTURAS DE DADOS

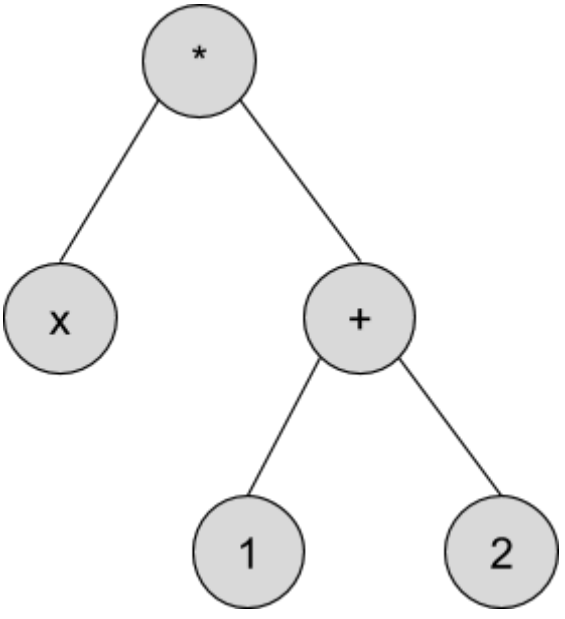
Laboratório 10 — Árvores Binárias

Inteligência & Tecnologia é uma companhia de tecnologia da informação multinacional. Essa companhia está desenvolvendo um novo *software* para ser adotado em suas calculadoras ***iCalc***. Uma das funcionalidades do *software* é otimizar expressões matemáticas informadas pelo usuário. Dessa forma, o objetivo do *software* é receber uma expressão aritmética inteira "$((4 * (5 - 2)) - (x + y))$" informada pelo usuário e devolver uma expressão equivalente otimizada "$(12 - (x + y))$" como saída. Maria está cursando Estruturas de Dados e foi contratada pela companhia para implementar essa nova funcionalidade utilizando conceitos de árvores binárias. Seu trabalho será otimizar expressões matemáticas.

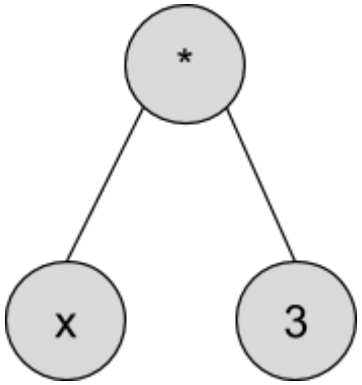
Tarefa

Neste laboratório, o programa deve receber uma expressão s como entrada e gerar uma árvore binária representando tal expressão. Após a construção da árvore, o programa deverá otimizar a árvore sempre que houver uma expressão constante em alguma subárvore. Finalmente, seu programa deve devolver a expressão otimizada. Veja o exemplo abaixo para a seguinte expressão "$(x * (1 + 2))$" informada pelo usuário:

Árvore Binária construída a partir da expressão.



Árvore Binária após o processo de otimização.



Expressão otimizada "$(x * 3)$".

Entrada

Expressão matemática s. As operações aritméticas permitidas são soma, subtração, multiplicação e divisão. As expressões conterão números negativos e positivos. Cada operação está rodeada por parênteses, mesmo quando não houver ambiguidade. Você pode supor que todas as variáveis são representadas por um único caractere e que parênteses, operadores e operandos estão separados por espaço.

Saída

O seu programa deve produzir uma única linha de saída contendo a expressão otimizada. Todos os casos de teste têm um espaço em branco no final do último parênteses. **Há uma quebra de linha no final da expressão.**

Exemplo

Entrada

```
( ( 4 * ( 5 - 1 ) ) - ( x + y ) )
```

Saída

```
( 16 - ( x + y ) )
```

Dicas

- Como cada elemento da expressão está separado por espaços, você pode ler string por string, usando `scanf("%s", buf)` e descobrir o tipo de cada elemento verificando os primeiros caracteres, por exemplo:

```
scanf("%s", buf);
/* início de alguma subexpressão */
if (buf[0] == '(') {
    ...
/* número */
} else if (buf[0] >= '0' && buf[0] <= '9') {
    valor = atoi(buf);
    ...
/* número negativo */
} else if (buf[0] == '-' && buf[1] >= '0' && buf[1] <= '9')
{
    valor = atoi(buf);
    ...
) else if (...) {
    ...
}
```

Critérios específicos

- Neste laboratório é obrigatório implementar e utilizar explicitamente uma árvore binária para representar a expressão lida do teclado e, em seguida, para otimizá-la.
- Para as turmas E e F, este laboratório tem peso 4.
- Para as turmas G e H, este laboratório tem peso 2.
- Submeter o seguinte arquivo:
    - lab10.c
    - arvore.c
    - arvore.h
- Tempo máximo de execução: 1 segundo.

Testando

Para compilar com o Makefile fornecido e testar um caso de teste:

```
make
./lab10 < arq01.in > arq01.out
diff arq01.res arq01.out
```

onde `arq01.in` é a entrada (casos de testes disponíveis no SuSy), `arq01.out` é a saída do seu programa e `arq01.res` é a solução provida para a entrada arq01.in. O Makefile também contém uma regra para testar todos os testes de uma vez; nesse caso, basta digitar:

```
make testar_tudo
```

Observações gerais

No SuSy, haverá 3 tipos de tarefas com siglas diferentes para cada laboratório de programação. Todas possuirão os mesmos casos de teste. As siglas são:

1. **SANDBOX:** Esta tarefa serve para testar o programa no SuSy antes de submeter a versão final. Nessa tarefa, tanto o prazo quanto o número de submissões são ilimitados, porém arquivos submetidos aqui **não serão corrigidos**.

2. **ENTREGA:** Esta tarefa tem limite de **uma única submissão** e serve para entregar a versão final dentro do prazo estabelecido para o laboratório. Não use essa tarefa para testar o seu programa: submeta aqui quando não for mais fazer alterações no seu programa.

3. **FORAPRAZO:** Esta tarefa tem limite de **uma única submissão** e serve para entregar a versão final após o prazo estabelecido para o laboratório, mas com nota reduzida (conforme a ementa). O envio nesta tarefa irá substituir a nota obtida na tarefa ENTREGA apenas se o aluno tiver realizado as correções sugeridas no feedback ou caso não tenha enviado anteriormente em ENTREGA.

Observações sobre SuSy:

- Versão do GCC: C-ANSI 4.8.2 20140120 (Red Hat 4.8.2-15).
- Flags de compilação:
  `-ansi -Wall -pedantic-errors -Werror -g -lm`
- Utilize comentários do tipo `/* comentário */`;
  comentários do tipo `//` serão tratados como erros pelo SuSy.

Além das observações acima, esse laboratório será avaliado pelos critérios gerais:

- Indentação de código e outras boas práticas, tais como:
  - uso de comentários (apenas quando forem relevantes);
  - código simples e fácil de entender;

    - sem duplicidade (partes que fazem a mesma coisa).

- Organização do código:
    - tipos de dados criados pelo usuário e funções bem definidas e tão independentes quanto possível.

- Corretude do programa:
    - programa correto e implementado conforme solicitado no enunciado;
    - eficiência do algoritmo (complexidade de tempo ou de espaço);
    - inicialização de variáveis sempre que for necessário;
    - desalocar toda memória alocada dinamicamente durante a execução do programa;
    - realizar leitura ou escrita em blocos de memória não alocados;
    - duplicidade de código;
    - má utilização de recursos (overhead de processamento ou de memória);
    - dentre outros critérios caso necessários.