Trabalho Prático 0: Notação Polonesa Reversa

Algoritmos e Estruturas de Dados III – 2017/1
Entrega: 20/04/2017

1 Introdução

João é o novo funcionário de uma empresa financeira. Nessa empresa ele é responsável por fazer as análises de investimentos, juros e rendimentos dos clientes, os ajudando a tomar as melhores decisões.

Como é comum no setor financeiro, o sistema da empresa utiliza a notação polonesa reversa para fazer todos os cálculos necessários. Dessa forma, o chefe do João pediu como primeira tarefa que ele aprendesse essa notação. A notação é simples e oferece muitos benefícios, João é esperto e aprendeu rapidamente, utilizando os exemplos mostrados na Tabela 1 e também alguns sites na Internet.

Notação Convencional	Notação Polonesa Reversa
$a + b$	$ab+$
$(a + b)/c$	$ab + c/$
$((a + b) * c)/(d - e)$	$ab + c * de - /$

Tabela 1: Exemplos de operações na Notação Polonesa Reversa.

O chefe de João ficou feliz com a rapidez de seu aprendizado e resolveu pedir para que ele resolvesse uma tarefa desafiante. Na semana anterior o sistema da empresa falhou e alguns arquivos com operações importantes salvas em notação polonesa reversa foram corrompidos. Felizmente, os arquivos ainda puderam ser abertos e apenas os operadores foram perdidos (todos eles!). Como exemplo, uma expressão em notação polonesa reversa que estava salva como 2 2 + = 4 mudou para 2 2 ? = 4 após o incidente. Para resolver o problema, o chefe do João pediu para que ele listasse, para cada operação, os possíveis operadores que poderiam gerar o resultado e o informou que originalmente nesses arquivos só existiam os operadores + e $*$, portanto ele não precisaria checar outros operadores. Para ilustrar e ajudar

Expressão original	Possível sequência de operadores
2 2 ? = 4	+, ∗
2 3 ? 6 ? = 11	++
2 3 ? 5 4 ? ? = 100	+∗∗

Tabela 2: Exemplos de expressões corrompidas e possíveis operadores.

o João a entender a tarefa, o chefe forneceu uma tabela com alguns exemplos (Tabela 2).

João ficou desesperado com a tarefa porque embora ele tenha entendido a notação, ele não sabe nada de computação. Dessa forma, você como um bom amigo do João, vai ajudá-lo, através da implementação de um código que receba uma expressão corrompida em notação polonesa reversa e retorne a lista de possíveis sequências de operadores.

2 Entrada e saída

A entrada é a padrão do C, stdin, não é necessário abertura de arquivos para leitura da entrada.

Entrada A entrada consiste de duas linhas. Na primeira linha, a expressão em notação polonesa reversa deve ser fornecida, os operadores a serem descobertos são representados pelo símbolo de interrogação (?) e todos os operandos e operadores são separados por um espaço. Na segunda linha, o resultado da expressão deve ser fornecido.

Exemplo de entrada

```
2 2 ? 2 2 ? 2 2 ? 2 2 ? ? ? ?
16
```

Saída A saída consiste de **todas** as sequências de operadores possíveis, uma sequência por linha. Note que **não** existe espaço entre os operadores em cada sequência.

Importante: a saída dever ser ordenada lexicograficamente e o operador + tem prioridade em relação ao operador ∗. Por exemplo, se existem duas sequências possíveis para uma dada expressão: +∗ e ++, a expressão ++ deve ser impressa primeiro.

Exemplo de saída

Para a entrada passada na seção 2, teremos a execução:

```
+++++++
+++*+++
++*++++
++**+++
+*+++++
+*+*+++
+**++++
+***+++
*++++++
*++*+++
*+*++++
*+**+++
**+++++
**+*+++
***++++
****+++
```

3 O que deve ser entregue

Deverá ser submetido um arquivo **.zip** contendo somente uma pasta chamada `tp0` e dentro desta deverá ter: (i) Documentação **em formato PDF** e (ii) Implementação.

Documentação Poderá ter no máximo 10 páginas e deverá seguir tanto os critérios de avaliação discutidos na Seção 4.1, bem como as diretrizes sobre a elaboração de documentações disponibilizadas no *moodle*.

Implementação Código fonte do seu TP (*.c* e *.h*)

Makefile Inclua um *makefile* na submissão que permita compilar o trabalho. É obrigatório o uso das *flags*: **-Wall -Wextra -Werror -std=c99 -pedantic** na compilação.

4 Avaliação

Eis uma lista **não exaustiva** dos critérios de avaliação que serão utilizados.

4.1 Documentação

Introdução Inclua uma breve explicação do problema que está sendo resolvido no seu trabalho e um resumo da sua solução.

Solução do Problema Você deve descrever a solução do problema de maneira clara e precisa, detalhando e justificando os algoritmos e estruturas de dados utilizados. Para tal, artifícios como pseudo-códigos, exemplos ou diagramas podem ser úteis. Note que documentar uma solução não é o mesmo que documentar seu código. **Não** é necessário incluir trechos de código em sua documentação nem mostrar detalhes de sua implementação, exceto quando estes influenciem o seu algoritmo principal, o que se torna interessante.

Análise de Complexidade Inclua uma análise de complexidade de tempo e espaço dos principais algoritmos e estrutura de dados utilizados. Cada complexidade apresentada deverá ser devidamente **justificada** para que seja aceita.

Avaliação Experimental Sua documentação deve incluir os resultados de experimentos que avaliem o tempo de execução de seu código em função de características da entrada. Cabe a você gerar entradas para esses experimentos. Por exemplo: se esse trabalho fosse sobre ordenação, seria interessante mostrar como o tempo de execução de cada algoritmo varia quando o número de items a serem ordenados aumenta. Para tal, um gráfico mostrando o tempo de execução em função do tamanho da entrada pode ser interessante. Você também deve interpretar os resultados obtidos. Comente sobre cada gráfico ou tabela que você apresentar mostrando o que é possível concluir a partir dele.

4.2 Implementação

Linguagem & Ambiente O seu programa deverá ser implementado na linguagem **C** e poderá fazer uso de funções da biblioteca padrão da linguagem. Trabalhos que utilizem qualquer outra linguagem de programação e/ou que façam uso de outras bibliotecas que não a padrão serão zerados. Além disso, certifique-se que seu código compile e funcione corretamente nas máquinas **Linux** dos laboratórios do DCC.

Casos de teste A sua implementação passará por um processo de correção automatizado, portanto, o formato da saída do seu programa deve ser idêntico

aquele descrito nessa especificação. Saídas com qualquer divergência serão consideradas erradas, mesmo que as divergências sejam *whitespaces*. e.g. espaços, *tabs*, quebras de linha, etc. Para auxilia-lo na depuração do seu código, será fornecido um pequeno, **não-exaustivo**, conjunto de entradas e suas respectivas saídas. É seu dever certificar-se que seu código funciona corretamente para qualquer entrada válida.

Alocação Dinâmica Algoritmos e estruturas de dados deverão fazer uso de memória alocada dinamicamente (`malloc()` ou `calloc()`). Certifique-se que seu programa utiliza essas regiões de memória corretamente, pois os monitores penalizarão implementações que realizam *out-of-bounds access* e que tenham vazamento de memória (não desalocar memoria dinâmica).
A alocação dinâmica deverá fazer uso das funções malloc() ou calloc() da biblioteca padrão C, bem como liberar tudo o que for alocado utilizando free(), para gerenciar o uso da memória. **DICA:** Utilize `valgrind` antes de submeter o seu TP.

Qualidade do código Seu código também será avaliado no quesito de legibilidade, dando atenção, porém não limitando-se, aos seguintes items: (i) **INDENTAÇÃO**; (ii) nomes de variável e função descritivos e claros; (iii) Modularização adequada; (iv) Comentários dentro de funções, explicando o que certos trechos mais complicados fazem; (v) Comentários fora de funções, explicando, em alto-nível, o que as funções mais importantes fazem; (vi) funções concisas que desempenham somente uma tarefa; (vii) **Proibido uso de variáveis globais**.

Atrasos Trabalhos poderão ser entregues após o prazo estabelecido, porém sujeitos a uma penalização regida pela seguinte fórmula:

$$\Delta_p = \frac{2^{d-1}}{0.32}\,\%$$

Por exemplo, se a nota dada pelo corretor for 70 e você entregou o TP com 4 dias corridos de atraso, sua penalização será de $\Delta_p = 25\%$ e, portanto, a sua nota final será: $N_f = 70 \cdot (1 - \Delta_p) = 52.2$. Note que a penalização é exponencial e 6 dias de atraso resultam em uma penalização de 100%.

5 Consideração Final

Assim como em todos os trabalhos dessa disciplina é estritamente proibida a copia parcial ou integra de códigos, seja da internet ou de colegas. Utilizare-

mos o algoritmo *MOSS* para detecção de plágio em trabalhos, seja honesto. Você não aprende nada copiando código de terceiros nem pedindo a outra pessoa que faça o trabalho por você. Se a cópia for detectada, sua nota será zerada e os professores serão informados para que as devidas providências sejam tomadas.

HAVE FUN!!!