Universidade Federal de Pernambuco
Centro de Informática
Bacharelado em Sistemas de Informação

IF969 Algoritmos e Estruturas de Dados — 2012.2

PRIMEIRA LISTA DE EXERCÍCIOS

- Esta lista contém **04** questões escritas e **02** questões práticas de programação.
- As respostas às questões escritas devem ser entregues ao professor em papel escrito à mão (à tinta e legível) até o final da aula do dia **14/Jan/13**.
- As questões práticas devem ser submetidas através do sistema de correção automática em `www.cin.ufpe.br/~if969`, seção "Listas" até o dia **13/Jan/13 às 17h** (hora do servidor)
- A lista é **individual**.
  - É permitido discutir as questões com os colegas e monitores mas as respostas devem ser escritas individualmente com suas próprias palavras.
  - É permitido estudar códigos de terceiros mas *não é* permitido copiar trechos substanciais e apresentá-los como trabalho seu.
  - A detecção de cópia implicará na atribuição de nota 0 (zero) à lista.
  - O professor poderá solicitar aleatoriamente a explicação oral de suas respostas até a divulgação das notas. Essa explicação deverá ser feita sem assistência. Esteja certo de ter compreendido suas respostas.

QUESTÃO 1 Eliminando repetições (1,5 pontos)

Escreva em pseudo-código um algoritmo que recebe como entrada um vetor V de n inteiros positivos e modifica-o de forma a eliminar os elementos repetidos. O algoritmo deve manter a mesma ordem dos valores originais porém deslocando elementos à esquerda de forma a ocupar o lugar de elementos repetidos eventualmente eliminados. As posições finais eventualmente livres devem ser preenchidas com zeros.

Exemplo: $V = (2, 4, 3, 9, 4, 2, 5, 8, 5) \rightarrow V' = (2, 4, 3, 9, 5, 8, 0, 0, 0)$.

Importante: O algoritmo só poderá usar, além do vetor de entrada, uma quantidade fixa de memória adicional, i.e., não é permitido criar um vetor ou lista auxiliar.

QUESTÃO 2 Pré-computando valores (1,5 pontos)

a) O cálculo de algumas funções matemáticas pode ser muito custoso. Assim, se um programa calcula muitas vezes, por exemplo, a função fatorial, pode ser útil manter uma tabela com valores pré-computados. Escreva em pseudo-código uma função *pcfatorial* que recebe um número $n \in \mathbb{N}$ e devolve como saída um vetor $F = (0!, 1!, 2!, \ldots, n!)$.

b) Um exemplo de tal programa seria um programa que calcula o *Triângulo de Pascal*:

$$\begin{array}{ccccccc} \binom{0}{0} & & & & & & \\ \binom{1}{0} & \binom{1}{1} & & & & & \\ \binom{2}{0} & \binom{2}{1} & \binom{2}{2} & & & & \\ \vdots & \vdots & \vdots & \ddots & & & \\ \binom{n}{0} & \binom{n}{1} & \binom{n}{2} & \cdots & \binom{n}{n} & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \end{array},$$

onde

$$\binom{a}{b} = \frac{a!}{b!(a-b)!}.$$

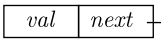
Escreva uma função *tpascal* em pseudo-código que recebe como entrada um valor $n \in \mathbb{N}$ e devolve como saída uma matriz $P_{(n+1)\times(n+1)}$ contendo as $n+1$ primeiras linhas do Triângulo de Pascal, i.e. $P[i,j] = \binom{i-1}{j-1}$, se $i \geq j$. **Importante:** Esta função deve usar a função *pcfatorial* do item (a) e a identidade $\binom{a}{b} = \binom{a}{a-b}$.

c) Suponha que tenhamos um jogo no qual fazemos $n \in \mathbb{N}$ tentativas independentes, cada uma com a mesma probabilidade de sucesso $p \in [0,1]$. A probabilidade de termos exatamente s sucessos nas n tentativas é dada pela função de probabilidade da chamada *distribuição binomial* como

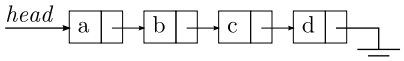
$$f(s;n,p) = \binom{n}{s} p^s (1-p)^{n-s}.$$

Escreva uma função *binom* que recebe como entrada três vetores $N = (n_1, \ldots, n_r) \in \mathbb{N}^r$, $P = (p_1, \ldots, p_r) \in [0,1]^r$ e $S = (s_1, \ldots, s_r) \in \mathbb{N}^r$ e devolve como resposta o vetor $B = (f(s_1;n_1,p_1), \ldots, f(s_r;n_r,p_r))$. **Importante:** Esta função deve usar a função *tpascal* do item (b) e pode assumir que $s_i \leq n_i$, para $i = 1, \ldots, r$.

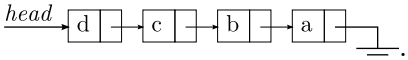
QUESTÃO 3 Invertendo uma lista (1,5 pontos)

Escreva um algoritmo em pseudo-código que recebe como entrada uma lista simplesmente encadeada L que utiliza a estrutura de nó habitual 

e que a modifica de forma a invertê-la. Por exmplo, a lista



deve ser convertida em



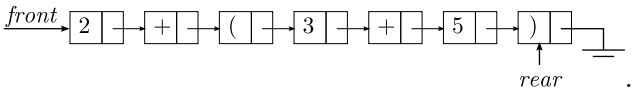
.

Importante: O algoritmo só poderá usar, além da lista de entrada, uma quantidade fixa de memória adicional, i.e., não é permitido copiar a lista para uma lista ou vetor auxiliar.

QUESTÃO 4 Expressões balanceadas (1,5 pontos)

Expressões aritméticas podem ser escritas com o auxílio de parênteses de forma a eliminar ambiguidades e especificar uma ordem precisa de execução das operações. Por exemplo, a expressão $10 + 3 \times 5 - 2$ pode ser avaliada de várias maneiras se não assurmirmos uma ordem de precedência. Porém, ao escrevermos $(10 + ((3 \times 5) - 2))$ a expressão deve ser inequivocamente avaliada como 23. Entretanto, para que esteja sintaticamente correta, a expressão deve ter *parênteses balanceados*, ou seja, cada "(" dever ter seu correspondente ")", e cada ")" deve sempre fechar um "(" anterior. Escreva em pseudocódigo um algoritmo que recebe como entrada uma expressão aritmética e devolve como resposta o valor verdadeiro (**V**), se a expressão estiver corretamente balanceada, ou falso (**F**), caso contrário.

Importante: A expressão de entrada vai estar codificada como uma lista de strings na qual cada elemento contém um número, um operador aritmético, ou um parênteses. Por exemplo, a expressão $2 + (3 + 5)$ será codificada como



.

Além dessa lista de entrada, o algoritmo só poderá usar mais uma pilha e mais uma quantidade fixa de variáveis que armazenam valores retirados da fila ou da pilha. Pode-se assumir como dadas apenas as funções *enfileirar*/*desenfileirar* e *empilhar*/*desempilhar*, sendo que não é permitido desenfileirar/desempilhar elementos de uma fila/pilha vazia.

QUESTÃO 5 (2 pontos)

Questão prática de implemetação. Ver Lista 1 em `www.cin.ufpe.br/~if969`.

QUESTÃO 6 (2 pontos)

Questão prática de implemetação. Ver Lista 1 em `www.cin.ufpe.br/~if969`.