IMD0029 - Estrutura de Dados Básicas 1 – 2017.2 – Prova 02
Prof. Eiji Adachi M. Barbosa

Nome:_____

Matrícula: _____

ANTES DE COMEÇAR A PROVA, leia atentamente as seguintes instruções:

- Esta é uma prova escrita de caráter individual e sem consultas a pessoas ou material (impresso ou eletrônico).
- A prova vale 7 de 10 pontos na segunda unidade e o valor de cada questão é informado no seu enunciado.
- Preze por respostas legíveis, bem organizadas e simples.
- As respostas devem ser fornecidas preferencialmente em caneta. Respostas fornecidas a lápis serão aceitas, mas eventuais questionamentos sobre a correção não serão aceitos.
- Celulares e outros dispositivos eletrônicos devem permanecer desligados durante toda a prova.
- Desvios éticos ou de honestidade levarão à anulação da prova do candidato (nota igual a zero).

Questão 1: (2,0 pontos) Considere uma lista duplamente encadeada com sentinelas cabeça (*head*) e cauda (*tail*) e cujos nós têm como conteúdo valores inteiros. Supostamente, esta lista deveria manter seus nós organizados em ordem crescente o tempo todo. No entanto, por algum defeito na implementação da função de inserir-ordenado, os nós da lista acabaram ficando desorganizados. Você olhou o conteúdo da lista e viu que eles estão completamente aleatórios. Você tentou reorganizar a lista de modo que seus nós ficassem em ordem crescente e, para isso, você implementou um Insertion Sort. No entanto, há tantos nós na lista que sua solução baseada no Insertion Sort ficou muito lenta. Analisando os dados armazenados na lista, você verificou que os nós desta lista assumem apenas os seguintes valores: 0, 1 ou 2. Daí você pensou: "Já sei! Eu não preciso de um algoritmo de ordenação completo. Eu consigo reorganizar esta lista com uma solução linear!". Neste contexto, faça uma função iterativa que reorganize esta lista, deixando-a em ordem crescente do valor dos nós. Sua função deverá obrigatoriamente ter complexidade assintótica **$\Theta(n)$** e deverá seguir a assinatura:

```
void List::Organize()
```

Obs.: Nesta questão, você não tem a sua disposição operações de inserção e remoção de nós da lista. Se necessário, você deverá criar seus próprios métodos, ou operar diretamente sobre os ponteiros dos nós.

Questão 2: (2,0 pontos) Considere uma lista duplamente encadeada com sentinelas cabeça (*head*) e cauda (*tail*). Faça uma função iterativa para remover os elementos repetidos desta lista e retornar quantos elementos foram removidos. Sua função deverá seguir a assinatura:

```
int List::RemoveRepeated()
```

Questão 3: (2,0 pontos): Na notação usual de expressões aritméticas, os operadores são escritos entre os operandos; por isso, a notação é chamada infixa. Na notação pós-fixa, ou notação polonesa inversa, os operadores são escritos depois dos operandos. Eis alguns exemplos:

Pós-Fixa	Infixa
AB+	(A+B)
AB+C*	((A+B)*C)
ABC+*	(A*(B+C))
ABC+*D/E-	(((A*(B+C))/D)-E)
AB*CD*+EF*GH*-/	(((A*B)+(C*D))/((E*F)-(G*H)))

Nesta questão, faça uma função que recebe como entrada uma string representando uma expressão aritmética em notação pós-fixa e retorna uma outra string representando esta mesma expressão aritmética em notação infixa. Sua função deverá obrigatoriamente usar uma pilha, ter complexidade assintótica **$\Theta(n)$** – 'n' é o tamanho da string de entrada – e seguir a assinatura:

```
string ConvertFromPostfixToInfix(string expression)
```

Obs.1: Considere que a expressão de entrada contém apenas letras representando os operandos da expressão e símbolos representando as quatro operações aritméticas básicas ('+', '-', '', '/'). Ou seja, cada caractere da string de entrada ou é um operando, representado por uma única letra, ou é um operador.*
Obs.2: Considere que a expressão de entrada sempre será bem formada. Ou seja, sua solução não precisa checar se ela está bem formada na notação pós-fixa.

Questão 4: (1,0 ponto) Para cada uma das afirmações a seguir, marque V (verdadeiro) ou F (falso), **justificando sucintamente** sua resposta (mesmo as verdadeiras devem ser justificadas). Marcações de V ou F **sem justificativas não serão aceitas.**

1 - ( ) De posse de uma lista simplesmente encadeada com ponteiros para início e fim, é possível criarmos uma estrutura que segue a estratégia F.I.L.O. (first-in last-out) da seguinte forma: ao realizarmos uma *inserção* nós inserimos no início da lista e ao realizarmos uma *remoção* nós removemos do fim da lista.

2 - ( ) De posse de uma lista simplesmente encadeada com ponteiros para início e fim, é possível criarmos uma estrutura que segue a estratégia F.I.F.O. (first-in first-out) da seguinte forma: ao realizarmos uma *inserção* nós inserimos no início da lista e ao realizarmos uma *remoção* nós removemos do fim da lista.

3 - ( ) As operações de inserir no início e no fim são realizadas em tempo constante tanto em listas simplesmente encadeadas quanto em listas duplamente encadeada.

4 - ( ) As operações de remover no início e no fim são realizadas em tempo constante tanto em listas simplesmente encadeadas quanto em listas duplamente encadeada.

5 - ( ) Ao implementarmos um Deque com base em arrays, não conseguimos prover as funções de inserir (pushBack e pushFront) e remover (popBack e popFront) em tempo constante.