IMD0029 - Estrutura de Dados Básicas 1 – Turma 04 – 2016.2 – Prova 01
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 10,0 pontos e o valor de cada questão é informado no seu enunciado.
- Preze por respostas legíveis, bem organizadas e simples.
- Leia atentamente às observações em cada questão. Elas fazem parte do enunciado.
- 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: (3,0 pontos) Considere uma lista duplamente encadeada, com sentinelas cabeça (*head*) e cauda (*tail*), conforme as estruturas a seguir:

```
List
{
      Node* head;
      Node* tail;
      Int quantity;
}
```

```
Node
{
      Node* next;
      Node* previous;
      int content;
}
```

Para esta estrutura lista, implemente uma função que recebe duas listas L1 e L2 já ordenadas em ordem crescente, possivelmente contendo elementos repetidos, e retorne uma nova lista L3 contendo todos os elementos de L1 e L2 ordenados em ordem decrescente. Além disso, a lista L3 não deverá conter elementos repetidos. A sua função deverá seguir a seguinte assinatura:

```
List Foo( List L1, List L2);
```

Abaixo seguem alguns exemplos de entradas e saídas esperadas para a sua função (na notação abaixo, os nós sentinelas estão omitidos para simplificar a representação):

```
Input: L1={ 1 <-> 4 <-> 5 <-> 6 <-> 8 }, L2={ 1 <-> 2 <-> 3 <-> 5 <-> 7 <-> 9 }
Output: L3={ 9 <-> 8 <-> 7 <-> 6 <-> 5 <-> 4 <-> 3 <-> 2 <-> 1}

Input: L1={ 1 <-> 1 <-> 2 <-> 4 }, L2={ 1 <-> 2 <-> 2 <-> 2 <-> 3 }
Output: L3={ 4 <-> 3 <-> 2 <-> 1 }
```

Obs.1: A sua solução deverá única e exclusivamente operar sobre listas encadeadas. Não tente usar arrays auxiliares.
Obs.2: Ao implementar a função `Foo`, você poderá reusar apenas as funções: `bool InsertBegin(Lista L, int x)` e `bool InsertEnd(Lista L, int x)`. Nenhuma outra função poderá ser reusada nesta questão, exceto aquelas que você mesmo implementar.
Obs.3: Tente implementar uma solução que tenha complexidade computacional de ordem linear em relação ao número de nós das listas de entrada. Soluções com complexidade superior à linear serão aceitas, mas a complexidade da solução fará parte do critério de correção.

Questão 2: (3,0 pontos) A Notação Polonesa Reversa (NPR) é uma notação matemática pós-fixa, isto é, uma notação em que os operadores seguem após os seus operandos. Por exemplo, a soma dos números 3 e 4 é expressa na NPR da seguinte forma "3 4 +", enquanto na notação convencional é expressa da seguinte forma "3 +

4". Similarmente, a expressão em notação convencional "3 – (4*2)" é expressa em NPR da seguinte forma: "3 4 2 * -". Nesta questão, escreva um programa que avalie uma expressão em NPR, retornando o valor resultante da expressão aritmética. Sua solução deverá seguir a seguinte assinatura:

```
int EvaluateExpression( string[] expression, int arraySize );
```

Abaixo seguem alguns exemplos de entradas e saídas esperadas para a função acima:

```
Input: 1 5 -
Output: -4

Input: 3 4 2 / -
Output: 1

Input: 3 4 * 2 / 6 -
Output: 0
```

Obs.1: Para resolver esta questão, sua solução deverá obrigatoriamente usar um dos seguintes tipos abstratos de dados: Pilha, Fila ou Deque. Pode considerar que todos estes TADs já estão implementados corretamente e podem ser reusados.
Obs.2: Considere que cada elemento do vetor de entrada `expression` corresponde a um elemento (operando ou operador) de uma expressão aritmética. Por exemplo, a expressão aritmética `"3 4 2 * –"` será representada como o seguinte vetor de strings: `{"3", "4", "2", "*", "–"}`. Além disso, considere que sua função sempre será chamada com vetores de string que representam expressões aritméticas em NPR que são válidas.
Obs.3: Para converter uma string contendo dígitos no seu respectivo valor inteiro use a função `int stoi( string )`.

Questão 3: (3,0 pontos) Considere uma lista simplesmente encadeada, sem sentinelas e que possui ponteiros para o seu início (*begin*) e o seu fim (*end*), conforme as estruturas a seguir:

```
List
{
      Node* begin;
      Node* end;
      Int quantity;
}
```

```
Node
{
      Node* next;
      Element content;
}
```

Nesta questão, implemente as funções *Inserir* e *Remover* para esta lista de modo que a sua implementação siga uma estratégia FIFO (*First-In First-Out*). Além disso, as funções de Inserir e Remover deverão obrigatoriamente ter complexidade computacional de ordem constante e seguir as seguintes assinaturas:

```
/* Insere um novo elemento na Lista */
Bool Inserir( Lista l, Elemento e);
```
(1,5 ponto)

```
/* Remove um elemento Da Lista */
Elemento Remover( Lista l );
```
(1,5 ponto)

Obs.1: Você deverá implementar as funções Inserir e Remover manipulando os nós da lista. Funções auxiliares só poderão ser usadas se você mesmo as implementar na questão. Em outras palavras, as funções típicas sobre uma lista não estão disponíveis para reuso nesta questão.

Questão 4: (1,0 ponto) O que são e para que servem os nós sentinelas em listas encadeadas?