Nome: _____ Matrícula: _____

Leia atentamente as instruções abaixo:

- Fazer o download do arquivo `Avaliacao2EDLab2-2019-3.zip` do *site* do curso e descompactá-lo na sua máquina. Este arquivo contém todos os códigos para o desenvolvimento da prova.
- A resposta de cada questão deve, **obrigatoriamente**, estar entre cada par de marcadores (//Qi, //-Qi). Assim, a questão 1 está entre //Q1 e //-Q1, a questão 2 entre //Q2 e //-Q2 e assim por diante. Não remover, em hipótese alguma, tais marcadores de questões da sua prova. Caso sua solução tenha mais de uma função ou operação, elas devem estar entre esses marcadores.
- Colocar no arquivo `main.cpp` seu nome completo e número de matrícula.
- A prova é **individual** e **sem qualquer tipo de consulta**.
- Existe apenas um projeto do Code::Blocks que será usado na prova.
- Antes de sair do laboratório, enviar ao servidor – usando a janela de upload – cada arquivo de código que contém as respostas das questões da sua prova. Aguarde um momento e verá as suas respostas de cada questão da prova.
- **O desenvolvimento e envio do código são de inteira responsabilidade do aluno!**
- Endereço do servidor: `http://172.18.40.97:8080/edlab2ufjf/`

Questões:

1. (25 Pontos) Considere o TAD `PilhaEncad` para representar uma **pilha encadeada** (detalhes deste TAD no projeto). Considere um vetor `vet` com `n` caracteres {, }, `[` ou `]`. Diz-se que `vet` é balanceado se para cada delimitador (chave ou colchete) aberto há outro do mesmo tipo que o fecha. Por exemplo `"{[][]}"` é balanceado, enquanto `"{[][]"` e `"{]"` não são balanceados. Implementar a função `bool balanceado(char vet[], int n)` que utiliza uma pilha de caracteres para testar se o vetor `vet`, contendo apenas {, }, `[` e `]` é balanceado (retorna `true`) ou não (retorna `false`). Usar uma pilha com a seguinte estratégia: enquanto percorre os caracteres do vetor:
   - se o caractere atual for { ou `[`, então empilhar o caractere;
   - se o caractere atual for } ou `]` e o topo for o seu par correspondente ({ ou `[`, respectivamente), então desempilhar;
   - qualquer outro caso significa que o vetor não está balanceado.

   O vetor está balanceado se, após a sua completa varredura, a pilha ficar vazia.

2. (25 Pontos) Considere o TAD `ListaCont` para representar uma **lista contígua** de valores inteiros (detalhes deste TAD no projeto). Implementar a operação `bool ListaCont::insereIntervalo(int a, int b)` que insere os elementos do intervalo fechado `[a,b]` no final da lista, **sem utilizar operações pré-definidas de inserção**. Se o vetor não tiver posições disponíveis, aumente a capacidade máxima do vetor para que caiba exatamente os elementos do intervalo, sem que nenhum dado atualmente representado na lista seja perdido. Ao final retornar se a capacidade máxima foi ou não aumentada.

3. (25 Pontos) Considere o TAD `ListaDupla` para representar uma **lista duplamente encadeada** de inteiros (detalhes deste TAD no projeto). Implementar a operação `removeMaior()` que remove o nó com maior valor de uma lista duplamente encadeada. A operação deve percorrer a lista uma única vez. Imprima uma mensagem de erro caso não seja possível remover o nó.

4. (25 Pontos) Considere o TAD `ListaEncad` que representa uma **lista simplesmente encadeada** de inteiros (detalhes deste TAD no projeto). Implementar a operação `void ListaEncad::removeBloco(int a, int b)` que remove todos os elementos localizados entre o elemento com valor `a` e o elemento com valor `b`, inclusive `a` e `b`. A operação deve percorrer a lista uma única vez. Assuma que: (i) o elemento de valor `a`, se houver, aparecerá antes do elemento com `b`; (ii) não havendo `a` na lista então nenhum elemento deve ser removido; e (iii) havendo um elemento com o valor `a` mas não havendo um com `b` então todos os elementos a partir de `a` devem ser removidos.