UNIVERSIDADE FEDERAL DE JUIZ DE FORA – INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

2ª avaliação de Estrutura de Dados e Laboratório de Programação II – 29/05/2017

Nome: _____ **Matrícula:** _____

Ler atentamente, por favor, as instruções abaixo:

- Fazer o download do arquivo Avaliacao2EDLab2-2017-1.zip do site do curso. Descompactá-lo em um diretório de 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 marcador (//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, obrigatoriamente.
- Colocar no arquivo `main.cpp` seu nome completo e matrícula.
- A prova é **individual** e **sem qualquer tipo de consulta**.
- Existe apenas um projeto do Code::Blocks que será usado na prova. **Fechar os outros programas durante a 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/ para submissão da prova

---

1) Dado o TAD `ListaCont`: lista contígua de números inteiros (ver os detalhes deste TAD no projeto em anexo). Pede-se para desenvolver a operação `contaValores()` para contar, e retornar, a quantidade de números ímpares maiores que `Val` existente na lista. Protótipo: (25)

```
int ListaCont::contaValores(int val);
```

---

2) Considerar o TAD `ListaSimplesOrd`: lista simplesmente encadeada de pontos 2D e ordenada pelo campo `y` do ponto – que é representado por dois números reais `x` e `y` (cada nó da lista possui dois números reais, as coordenadas de um ponto). Desenvolver a operação `inserir()` no MI do TAD para inserir um novo nó (novo ponto) na lista ordenada de ponto de forma que a lista permaneça ordenada pela coordenada `y`. Protótipo: (25)

```
void ListaSimplesOrd::inserir(float x, float y);
```

---

3) Considerar o TAD `ListaDupla`: lista duplamente encadeada com descritor de números reais (ver os detalhes deste TAD no projeto em anexo). Desenvolver a operação `particiona()` no MI do TAD para particionar uma lista em duas, segundo um valor dado `val`. Todos os nós da lista até encontrar a primeira ocorrência do nó com valor `val` devem permanecer na lista original (inclusive o nó com o próprio `val`) e a segunda parte – o restante dos nós da lista original – em uma nova lista cujo ponteiro deve ser retornado pela operação. Se não houver nenhuma ocorrência de `val` na lista, retornar lista vazia. Protótipo: (25)

```
ListaDupla* ListaDupla::particiona(float val);
```

---

4) O conteúdo de um vetor **vet** com **n** caracteres **a**, **b** e **c** deve satisfazer o padrão: a quantidade de **a**'s é igual a quantidade de **b**'s que, por sua vez, é igual a quantidade de **c**'s. Sendo assim, por exemplo, **acbcab**, **aabbcc** e **aacabcbbc** são cadeias que satisfazem o padrão, mas **ab**, **aacbb** e **caab** não satisfazem-no. Desenvolver uma função de PA para retornar **true** se **vet** satisfaz o padrão e **false** caso contrário (ver protótipo). Para tanto, deve-se usar 2 pilhas (**p1** e **p2**) e a função não pode usar nenhum contador de caracteres e tampouco o tamanho das pilhas. Considerar a seguinte estratégia para verificar se **vet** satisfaz, ou não, o padrão. Seja *x* (**a**, **b** ou **c**) o caractere atual do vetor **vet** (apenas uma ação com testes nessa ordem deve ser feita): (25)

i. se ambas as pilhas estiverem vazias, empilhar *x* sobre o topo de uma delas;
ii. se o topo de uma das pilhas for *x*, empilhar *x* sobre o topo;
iii. se uma das pilhas estiver vazia, empilhar *x* na pilha vazia;
iv. se *x* for diferente do topo de ambas as pilhas, desempilhar o topo de ambas.

O vetor **vet** satisfaz o padrão se, após a sua completa varredura – caractere por caractere –, as 2 pilhas tornarem-se vazias. Protótipo:

```
bool verificaPadrao(char *vet, int n);
```

BOA PROVA