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

3ª avaliação de Estrutura de Dados e Laboratório de Programação II – 25/11/2019

Nome: _____ **Matrícula:** _____

Ler atentamente, por favor, as instruções abaixo:

- Fazer o download do arquivo **Avaliacao3EDLab2-2019-3.zip** do *site* do curso e descompactá-lo em 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). Dessa forma, a questão 1 está entre //Q1 e //-Q1, a questão 2 entre //Q2 e //-Q2 e assim por diante. Não remover, em hipótese alguma, estes marcadores de início e fim 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 número de matrícula.
- A prova é **individual** e **sem qualquer tipo de consulta**.
- Há apenas um projeto do `Code::Blocks` 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 para ver as suas respostas de cada questão da prova.
- **O desenvolvimento e envio do código são de inteira responsabilidade do aluno!**
- Para o envio da prova, usar o servidor: `http://172.18.40.97:8080/edlab2ufjf/`.

---

1) Desenvolver uma operação para uma árvore binária (AB) que calcula estas duas quantidades de forma independente (variáveis separadas):
   - o número de nós com valores absolutos impares; e
   - o número de nós com 0 (zero) ou 2 (dois) filhos.

   Esta operação deve percorrer a árvore binária APENAS UMA ÚNICA VEZ fazendo um percurso em pré-ordem. Cuidado com a inicialização e atualização dos parâmetros passados por referência. Protótipo: [30]

```
void ArvBinBusca::nosImpares02Filhos(int *nImpar, int *n2Filhos);
```

---

2) Desenvolver uma operação **NÃO RECURSIVA** para inserir um novo nó com valor val em uma árvore binária de busca. Caso já exista algum nó com o valor val, não inserir um novo nó e emitir uma mensagem. Protótipo: [30]

```
void ArvBinBusca::insNaoRec(int val);
```

---

3) Dados um intervalo fechado $I = [a, b]$ e uma árvore binária de busca (ABB), desenvolver uma operação para alocar, retornar e preencher um vetor com os valores de todos os nós do nível k da ABB e que estão no intervalo I. Além disso, a operação deve preencher o vetor com os valores dentro do intervalo I em ordem crescente. O tamanho do vetor (capacidade máxima) deve ser igual a 1 mais o número máximo de nós da ABB no nível k. Caso haja folga no vetor (região do final do vetor não usada para armazenar qualquer valor da ABB), ela deve ser preenchida com o valor -1. Considerar que $a \leq b$ e, obrigatoriamente, usar a propriedade fundamental da ABB para visitar o número mínimo de nós. Não usar nenhum método de ordenação de vetores. Protótipo: [40]

```
int* ArvBinBusca:: vetIntervalo(int a, int b, int k);
```
