**UFJF / DCC Estrutura de Dados e Lab. de Programação II - TVC 3 25/07/2016**

Nome: _____ Matrícula: _____

**Leia atentamente as instruções abaixo:**

- Fazer o download do arquivo `Avaliacao3EDLab2016.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, elas devem estar entre esses marcadores, obrigatoriamente.
- Colocar no arquivo `main.cpp` seu nome completo e a turma tanto de ED e Lab. II.
- Antes de sair do laboratório, enviar para o servidor - usando a janela de upload - cada arquivo de código que contém as respostas das questões da sua prova.
- Existe apenas 1 projeto do CodeBlocks com o TAD que será usado na prova.
- **O desenvolvimento do código é de inteira responsabilidade do aluno!**

---

1. (30 Pontos) Dada uma árvore binária, desenvolver uma operação para verificar se esta representa uma árvore **estritamente** binária, isto é, uma árvore onde cada nó possui apenas 0 ou 2 filhos.

```
bool ArvBinBusca::ehEstritamente();
```

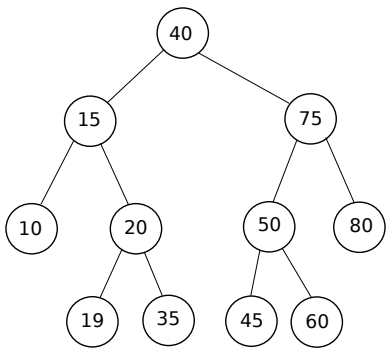
2. (35 Pontos) Implemente uma operação que dado um valor inteiro `ch`, procura e imprime o valor mais próximo dessa chave (incluindo a possibilidade do valor ser igual a `ch`) contido na árvore binária de busca. A função deve imprimir o valor do nó e a diferença entre a informação desse nó e a chave. Se a chave existir na árvore essa função deve imprimir também a mensagem *encontrado*. Dica: se necessário, utilize a função `abs()` da linguagem C/C++ para obter o valor absoluto de um número inteiro.

```
void ArvBinBusca::maisProximo(int ch);
```

3. (35 Pontos) Dada uma árvore binária de busca e uma chave `ch`, implementar uma operação que aloca (com um número de elementos apropriado), preenche e retorna um vetor `int *vet` com os valores armazenados nos nós que formam o caminho para encontrar (ou não) a chave `ch`. Se a chave for encontrada, completar as posições não utilizadas do vetor com o valor $-1$. Se a árvore for vazia a função deve retornar `NULL`. A função deve armazenar em `num` o tamanho do vetor que foi alocado.

```
int * ArvBinBusca::criaVetCaminho(int ch, int *num);
```

*Exemplo*:




*Saída*:

```
Questão 1: arv.ehEstritamente()
 Saída = 1 (true)
Questão 2: arv.maisProximo(63)
 valor = 60, diferenca = 3
 arv.maisProximo(20)
 encontrado, valor = 20, diferenca = 0
Questão 3: v = arv.criaVetCaminho(45,&n)
 v = [ 40 75 50 45 ]
 arv.criaVetCaminho(8)
 v = [ 40 15 10 -1 ]
```