UFJF / DCC Estrutura de Dados e Lab. de Programação II - TVC 3 07/03/2016

Nome: _____ Turma ED: _____ Turma LabII: _____

Leia atentamente as instruções abaixo:

- Fazer o download do arquivo `Avaliacao3EDLab2015.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.
- Se for preciso implementar funções auxiliares para resolver uma determinada questão, implemente-as dentro dos marcadores da respectiva questão.
- 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. Aguarde um momento e verá as respostas de cada questão 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) Dado uma árvore binária de busca, desenvolver uma operação para calcular e retornar o número de nós da árvore que estão no intervalo fechado $[a, b]$. Percorrer a árvore levando-se em consideração a propriedade de árvore binária de busca.

```
int ArvBinBusca::contaIntervalo(int a, int b);
```

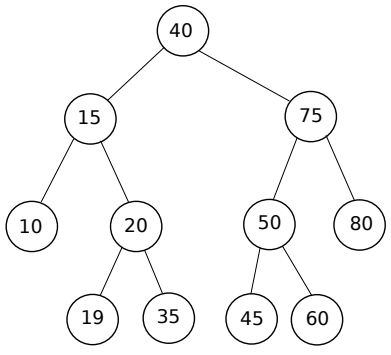
2. (35 Pontos) Implemente a operação `buscaPai(int x)` que dado um valor inteiro `x` faça a busca na árvore binária de busca pelo nó com o valor `x` e retorne um ponteiro para o seu pai. Verifique se sua implementação está correta usando esta operação no programa principal (`main.cpp`) e imprima o valor do nó que é pai de `x`.

```
No* ArvBinBusca::buscaPai(int x);
```

3. (35 Pontos) Dada uma árvore binária de busca, criar, preencher e retornar um vetor `int *vet` contendo os valores armazenados do nível k da árvore binária de busca. Para alocar o vetor lembre-se da propriedade de árvores binárias sobre o número de nós (máximo) em um determinado nível `k`.

```
int* ArvBinBusca::criaVetNivel(int k);
```

Exemplo:



Saída:

```
Questão 1: arv.contaIntervalo(20,50)
 Saída = 5

Questão 2: arv.buscaPai(60)
 Retorna ponteiro para o nó 50

Questão 3: arv.criaVetNivel(2)
 vet = [ 10 20 50 80 ]
```