UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Faculdade da Computação

Trabalho de AED2 – Valor 15 Pontos

Prof. Luiz Gustavo Almeida Martins

- Pode ser realizado em grupo de **no máximo 3 alunos**.
- Deve ser entregue em mídia (ex: DVD) ATÉ o dia **27/07/2017**.
- **Apresentações individuais** dos membros do grupo ocorrerão **entre os dias 28/07 a 01/08** (cada grupo deve **agendar seu horário previamente com o professor**).
- Os códigos deverão ser implementados em Linguagem C e utilizar os TADs implementados pelos alunos (está vetado o uso de bibliotecas de árvore prontas).

Parte 1 – Árvore Genérica

Implemente uma árvore genérica de inteiros, utilizando a estrutura de representação dinâmica (campos informação, filho à esquerda e irmão à direita). Além das operações básicas vistas em sala de aula, o TAD também deve contemplar as seguintes operações:

- **void preorder(Arv A):** percorre a árvore na pré-ordem, apresentando cada elemento visitado.
- **void postorder(ArvA):** percorre a árvore na pós-ordem, apresentando cada elemento visitado.
- **void percorre_nivel(Arv A):** percorre a árvore em largura (por nível), apresentando cada elemento visitado.
- **int nro_derivacao(Arv A):** retorna a quantidade de nós de derivação de uma árvore genérica.
- **int grau_arv(Arv A):** retorna o grau da árvore genérica dada como entrada.
- **int qtde_nos(Arv A, int grau):** retorna a quantidade de nós da árvore que possui o grau dado como entrada.
- **int altura_no(Arv A, int elem):** retorna a altura do nó da árvore genérica cujo valor é igual a elem. Se não existir tal nó, a função deve retornar -1.
- Implementar um programa de usuário que forneça um menu para acesso às funcionalidades acima.

Parte 2 – Árvore Binária de Busca (ABB)

Implemente uma ABB cujo registro é formado pelo nome, idade (campo chave) e curso dos alunos de uma Universidade. Além das operações básicas vistas em sala de aula, o TAD também deve contemplar as seguintes operações:

- **reg * maior(Arv A):** retorna o aluno mais velho cadastrado. Se houver repetição, deve ser mostrada apenas a 1ª ocorrência.
- **int de_maior(Arv A):** retorna a quantidade de alunos que são maiores de idade (idade ≥ 18).
- **int qtde_nos(Arv A, int ini, int fim):** retorna a quantidade de alunos que estão entre as idades mínima (ini) e máxima (fim) fornecidas como entrada.
- **int um_filho(Arv A):** retorna a quantidade de nós da ABB que possuem apenas um filho.
- **int completa(Arv A):** verifica se a ABB é completa ou não.
- **int altura_arv(Arv A):** retorna a altura da ABB fornecida como entrada.

- **Arv juntar(Arv A1, Arv A2):** recebe duas ABBs como entrada e retorna uma 3ª formada pela intercalação das 2 primeiras. Lembre-se que a propriedade de ordenação de uma ABB deve ser mantida.
- Implementar um programa de usuário que forneça um menu para acesso às funcionalidades acima.

Parte 3 – Árvore AVL

Implemente uma árvore AVL cujo registro é formado pelo identificador do ponto de rede, localização, capacidade e tráfego (campo chave) de uma rede de dados. Além das operações básicas vistas em sala de aula, o TAD também deve contemplar as seguintes operações:

- **reg * menor_trafego(Arv A):** retorna o ponto de rede com menor tráfego. Se houver repetição, deve ser mostrada apenas a 1ª ocorrência.
- **int qtde_ocioso(Arv A):** retorna a quantidade de pontos que estão usando menos da metade de sua capacidade.
- **int nro_folha(Arv A):** retorna a quantidade de nós folha de uma árvore AVL.
- **int cheia(Arv A):** verifica se a árvore AVL de entrada é cheia ou não.
- **int nivel_no(Arv A, reg elem):** retorna o nível (profundidade) do nó de uma árvore AVL cujo valor é igual a elem. Se não existir tal nó, a função deve retornar -1.
- **int iguais(Arv A1, Arv A2):** compara duas árvores binárias e indica se elas são iguais ou não.

- Implementar um programa de usuário que forneça um menu para acesso às funcionalidades acima.