Lista de Exercícios de CES-11 / 2010
CTA - ITA - IEC

Conteúdo: Pilhas, Filas, Deques, Árvores
Importante: Não vale nota, ou seja, não é preciso entregar!

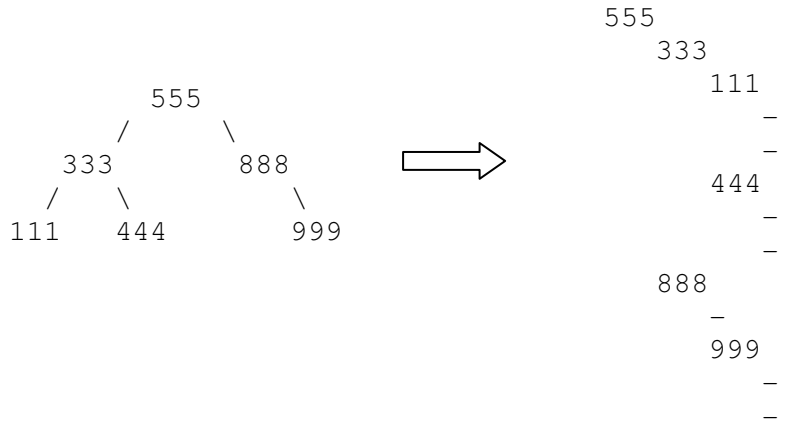
Nas questões abaixo, considere a seguinte estrutura de árvore binária:

```
struct node {
              int valor;
              node *esq, *dir;
            }
```

1. Para cada item abaixo, escreva uma função que recebe como parâmetro o ponteiro para a raiz de uma árvore binária.
   a) Imprimir o seu percurso pré-ordem.
   b) Imprimir o seu percurso in-ordem.
   c) Imprimir o seu percurso pós-ordem.
   d) Imprimir apenas o valor do primeiro e do último nó do seu percurso in-ordem.
   e) Calcular a quantidade de nós dessa árvore.
   f) Transformar essa árvore no seu "espelho", sem fazer novas alocações. (trocar todo ramo direito pelo ramo esquerdo)
   g) Verificar se um determinado valor k (também recebido como parâmetro) está ou não presente na árvore. Obs.: Considere que a árvore não é de busca.
   h) Verificar se essa árvore é ou não de busca.

2. Dados os ponteiros para a raiz de uma árvore binária e para um determinado nó x dessa mesma árvore, escreva uma função que retorne a profundidade de x na árvore.

3. Escreva uma função que imprima os valores armazenados em uma árvore binária com recuos de margem proporcionais à profundidade do nó na árvore. Imprima '-' para representar **`null`**. Veja o exemplo abaixo:



4. Dada uma árvore binária de busca sem repetição de valores, escreva uma função que receba o ponteiro para a raiz dessa árvore e um valor x, presente na árvore, e devolva o sucessor de x nessa mesma árvore.

5. Implemente as funções de navegação em uma árvore N-ária baseada em vetores de apontadores (defina um valor máximo para N). As funções são: eh_vazia(árvore), eh_raiz(nó), eh_caçula(nó), eh_folha(nó), primeiro_filho(nó), pai(nó), irmão_direito(nó).

6. Implemente uma forma de determinar, numa árvore binária, o ancestral comum de maior nível de dois nós fornecidos.

7. Implemente uma função para criar uma árvore binária de busca balanceada dado um vetor ordenado com todos os elementos que devem fazer parte da árvore.

8. Considere o algoritmo de inserção numa árvore de busca binária não balanceada. Dada uma seqüência de números de 1 a 15, em que ordem devo inserir esses números para que a árvore resultante esteja balanceada (mínima altura)? Como devo inserir os números para obter a árvore com altura máxima?

9. Considere uma árvore de busca binária completa com os números de 1 a 15 associados aos seus nós. Considere o algoritmo de remoção sem balanceamento. Realize a remoção dos números pares, iniciando do 2 até o 14.

10. Considere um vetor com uma seqüência de N caracteres distintos (ABCD....). Esses caracteres são inseridos nesta ordem numa estrutura de dados. (a) Qual o resultado de fazer N pushes seguidos de N pops desses caracteres numa pilha? (b) E se inserir os N cara teres e depois retira-los de uma fila? (c) O que aconteceria se para cada push for feito um pop imediatamente, N vezes? (d) O que acontece se trocar a ordem das operações de inserção e de remoção na fila? (e) É possível gerar qualquer permutação da seqüência de caracteres utilizando uma seqüência de pushes e pops numa pilha? (tente com três caracteres).

11. Implemente as operações de deque em uma lista circular duplamente ligada com nó líder.

12. Faça a análise aproximada da complexidade das duas implementações abaixo para calcular os números de Fibonacci e as compare:

```
int fibo1(int n)
{
    if (n<=1) return 1;
    else return fibo1(n-1)+fibo1(n-2);
}
```

```
int _aux_fib(int n, int a, int b)
{
    if (n==0) return a;
    else return _aux_fib(n-1, b, a+b);
}

int fibo2(int n)
{
    return _aux_fib(n, 1, 1);
}
```