Árvore Binária e Árvore Binária de Busca

Lista de Exercícios

Considere a seguinte declaração para os nós de uma árvore binária:

```
typedef struct no
{
        int v;
        struct no *esq, *dir;
} No;
```

1. Escreva uma função que realiza um percurso em largura na árvore. Você pode usar uma fila auxiliar.

```
void largura(No* t);
```

2. Escreva uma função recursiva que conta o número de folhas em uma árvore binária.

```
int conta folhas(No* t);
```

3. Escreva uma função recursiva que verifica se um valor v está presente na árvore t (considere que não há nenhuma garantia a respeito da ordem dos valores na árvore).

```
int busca(No* t, int v);
```

4. Escreva uma função espelho(t) que retorna uma nova árvore t, mas com as sub-árvores esquerda e direita de todos os nós trocadas. A árvore original não deve ser alterada.

```
No* espelho(No* t);
```

5. Em uma árvore de expressão, um nó que é um operando (uma letra entre a e z) não deve ter filhos e um nó que é um operador (+, -, * ou /) deve ter duas sub-árvores não vazias que correspondem a duas expressões válidas. Escreva uma função que verifica se uma árvore de expressão é válida, considerando também que uma árvore vazia é inválida e que a presença de quaisquer outros caracteres também tornam uma árvore inválida.

```
int expr_valida(ap_no t);
```

6. Escreva a árvore binária correspondente à seguinte expressão: (a+b) * (c−d)−e * f. Escreva os elementos da árvore considerando os seguintes percursos pré-ordem e pós-ordem.

7. Seja bal o fator de balanceamento de um nó dado pela fórmula hd − he, onde he é a altura da sub-árvore esquerda e hd é a altura da sub-árvore direita. Escreva uma função recursiva que calcula o bal de todos os nós da árvore.

```
typedef struct no
{
        int v;
        int bal;
        struct no *esq, *dir;
} No;
```

```
void calcula bal(No* t);
```

Seria eficiente utilizar uma função auxiliar altura para calcular os fatores de balanceamento?

Caso você ache que não, escreva uma solução mais eficiente.

8. Dada uma árvore com fatores de balanceamento calculados de acordo com a definição anterior, escreva uma função recursiva que retorna o valor máximo de bal na árvore.

```
int max bal(No* t);
```

9. Escreva uma função não-recursiva que recebe uma árvore binária de busca t como parâmetro e retorna o apontador para o nó cuja chave possui o valor mínimo ou NULL caso árvore esteja vazia.

```
No* minimo(No* t);
```

10. Escreva uma função não-recursiva que verifica a existência de algum elemento com chave negativa na árvore.

```
int existe chave negativa(No* t);
```

11. Escreva uma função que verifica se uma árvore de busca é válida.

```
int verifica busca(No* t);
```

12. Considerando os algoritmos de inserção e remoção, desenhe a árvore binária de busca resultante da inserção dos seguintes elementos (nesta ordem): 20, 25, 10, 5, 12, 22 e 23. Remova a raiz da árvore. Quais elementos podem ser utilizados para substituir a raiz?

13. Escreva uma função que percorre uma árvore de busca e retorna uma lista ligada ordenada contendo todos os elementos da árvore. A sua função deve ter complexidade proporcional ao número de nós da árvore (O(n)). Você pode usar funções auxiliares.

```
typedef struct no_lista
{
        int chave;
        struct no_lista* prox;
} No_lista;

No_lista* lista(No* t);
```