Estruturas de Dados e Seus Algoritmos
Lista de Exercícios - Árvores Binárias e Árvores Binárias de Busca

Dada a seguinte representação de uma árvore binária:

```
typedef struct ab {
     int info;
     struct ab *esq, *dir;
} TAB;
```

Responda às seguintes questões

1. Copiar uma árvore binária: `TAB* copia(TAB *a);`
2. Escreva uma função em C que faz o espelho de uma árvore binária (o que está à esquerda na árvore original, estará a direita no espelho, e vice-versa): `TAB* espelho(TAB *a);`
3. Escreva uma função em C que, dadas duas árvores deste tipo, testa se estas árvores são iguais. A função retorna um se elas são iguais e zero, caso contrário. A função deve obedecer ao seguinte protótipo: `int igual(TAB *a1, TAB* a2);`
4. Retornar o maior elemento da árvore binária: `int maior(TAB *a);`
5. Testar se uma árvore é zigue-zague, isto é, todos os nós internos possuem exatamente uma sub-árvore vazia: `int zz(TAB *a);`
6. Escreva uma função em C que, dada uma árvore binária qualquer, retire todos os elementos pares da árvore original. A função deve ter o seguinte protótipo: `TAB* retira_pares(TAB* arv);`
7. Retornar todos os ancestrais de um nó na árvore de busca binária, da raiz da árvore até o elemento passado como parâmetro, usando a biblioteca de lista encadeada: `TLista* ancestrais(TAB *a, int elem);`
8. Escreva uma função em C que, dada uma árvore binária de busca qualquer, retorne, num vetor, todos os elementos menores que N. A função deve ter o seguinte protótipo: `int* mN(TAB *a, int N);`
9. Suponha que a estrutura TAB tenha um campo cor (int cor). Escreva uma função em C que, ao receber uma árvore binária “sem cor” e totalmente balanceada (isto é, a distância da raiz a qualquer folha da árvore é sempre a mesma), retorne esta árvore com os nós coloridos somente de preto e branco, sendo que o nó pai NUNCA pode ter a mesma cor de seus filhos. A função deve possuir o seguinte protótipo: `void colore(TAB* arv);`