```
#include <stdio.h>
#include <malloc.h>
#define true 1
#define false 0

typedef int bool;
typedef int TIPOCHAVE;

typedef struct aux {
    TIPOCHAVE chave;
    struct aux *esq;
    struct aux *dir;
    int bal;
} NO, *PONT;

/* cria um novo (aloca memoria e
preenche valores) no com chave=ch e
retorna
       seu endereco */
PONT criarNovoNo(TIPOCHAVE ch){

}


// Retorna o maior valor entre dois
inteiros
int max(int a, int b){

}
```

```
// Retorna a altura de uma (sub-)arvore
int altura(PONT p){

}


/* Exibe arvore Em Ordem         */
void exibirArvoreEmOrdem(PONT raiz){

}

/* Exibe arvore Pre Ordem         */
void exibirArvorePreOrdem(PONT raiz){

}

/* Exibe arvore Pos Ordem         */
void exibirArvorePosOrdem(PONT raiz){

}

/* Exibe arvore Em Ordem (com parenteses
para os filhos)    */
void exibirArvore(PONT raiz){

}

/* Exibe arvore Pre-Ordem indicando pai
de cada no    */
```

```
void exibirArvore2(PONT raiz, TIPOCHAVE
chavePai){

}


// Verifica se árvore é AVL
bool ehAVL(PONT p){

}


// Atualiza o balancemento total
int atualizarBalanceamentoTotal(PONT
raiz){

}


/* Rotações à direita (LL e LR)
   Retornará o endereço do nó que será a
nova raiz da subárvore originalmente
   iniciada por p */
PONT rotacaoL(PONT p){


}

/* Rotações à esquerda (RR e RL)
   Retornará o endereço do nó que será a
nova raiz da subárvore originalmente
```

```
   iniciada por p */
PONT rotacaoR(PONT p){

}



/* Inserção AVL: p é inicializado com o
endereco do nó raiz e
   *alterou com false
*/
void inserirAVL(PONT* pp, TIPOCHAVE ch,
bool* alterou){


}


/* retorna o endereco do NO que contem
chave=ch ou NULL caso a chave nao seja
       encontrada. Utiliza busca binaria
recursiva
*/
PONT buscaBinaria(TIPOCHAVE ch, PONT
raiz){


}
```

```
// Busca binária não recursiva
devolvendo o nó pai
PONT buscaNo(PONT raiz, TIPOCHAVE ch,
PONT *pai){


}

/* Auxilir da funcao excluirChave,
procura a maior chave menor que a chave
que
   serah excluida            */
PONT maiorAEsquerda(PONT p, PONT *ant){

}


/* exclui a chave com valor igual a ch
*/
bool excluirAVL(PONT* raiz, TIPOCHAVE
ch, bool* alterou){

}




/* funcao auxiliar na destruicao
(liberacao da memoria) de uma arvore */
void destruirAux(PONT subRaiz){
```

```
}

/* libera toda memoria de uma arvore e
coloca NULL no valor da raiz    */
void destruirArvore(PONT * raiz){

}


//inicializa arvore
void inicializar(PONT * raiz){

}
```