André Luiz Teixeira Kohlrausch 201702725

Tema: Árvore AVL

1. Implemente e teste uma função que retorne a quantidade de nós de uma árvore AVL que possuem apenas um filho.

```c
int filho_unico(ArvAVL *raiz){

    static int quantidade = 0;
    int filho_d, filho_e;
    if(raiz == NULL)
        return 0;
    if(*raiz != NULL){

        filho_d = 0;
        filho_e = 0;

        if(((*raiz)->esq) == NULL){
            filho_e = 1
        }

        if(((*raiz)->dir) == NULL){
            filho_d = 1;
        }

        if((filho_d + filho_e) == 1){
            quantidade++;
        };

        filho_unico(&((*raiz)->esq));
        filho_unico(&((*raiz)->dir));

        return quantidade;
    }
}
```

2. Implemente e teste uma função (recursiva ou não recursiva) que verifique a existência de um valor X numa árvore AVL. Caso não se encontre o valor x, deve-se retornar, por referência, os valores imediatamente inferior e superior ao valor x, se houver. Na função main, imprimir uma das seguintes mensagens: a) “O valor de x encontra-se na AVL.” ou b) “Limite inferior de x encontrado na AVL: <limite_inferior>.” e/ou “Limite superior de x encontrado na AVL: <limite_superior>.”
Protótipo da função: *int consultaIntervalarAVL(ArvAVL *raiz, int x, int *limInf, int *limSup);*

```c
#include <stdlio.h>
#include <stdlib.h>
#include "ArvoreAVL.h"

int main(){
    ArvAVL* avl;
    int * lim_inf, * lim_sup;
    int res, i, valor, resultado;
    int N = 4, dados[4] = {2,300,-5,3};

    avl = cria_ArvAVL();

    for(i=0;i<N;i++){
        res = insere_ArvAVL(avl,dados[i]);

    }

    lim_inf = (int *)malloc(sizeof(int));
    lim_sup = (int *)malloc(sizeof(int));

    scanf("%d",&valor);

    resultado = consultaIntervalarAVL(avl,valor,lim_inf, lim_sup);

    if(resultado == 1){
        printf("O valor de %d encontra-se na AVL.\n",valor);
    }
    else if(resultado == 2){
        printf("Limite superior de %d encontrado na AVL: %d.\
n",valor,*lim_sup);
    }
    else if(resultado == 3){
        printf("Limite inferior de %d encontrado na AVL: %d.\
n",valor,*lim_inf);
    }
    else if(resultado == 5){
        printf("Limite superior de %d encontrado na AVL: %d.\
n",valor,*lim_sup);
        printf("Limite inferior de %d encontrado na AVL: %d.\
n",valor,*lim_inf);
```

```
    }


    libera_ArvAVL(avl);
    free(lim_inf);
    lim_inf = NULL;
    free(lim_sup);
    lim_sup = NULL;

    return 0;
}

/*
 * Código da função

int consultaIntervalarAVL(ArvAVL *raiz, int x, int *limInf, int *limSup){

    int resultado, contador;

    resultado = consulta_ArvAVL(raiz, x);

    if(resultado == 1){
        return resultado;
    }
    else{
        *limInf = *limSup = 0;

        for(contador = x + 1;contador < x * 100; contador++){
            //busca o limite superior
            if(consulta_ArvAVL(raiz,contador) == 1){
                *limSup = contador;
                resultado += 2;
                break;
            }
        }
        for(contador = x - 1;contador > x * (-100); contador--){
            //busca o limite superior
            if(consulta_ArvAVL(raiz,contador) == 1){
                *limInf = contador;
                resultado += 3;
                break;
            }
        }
    }
    return resultado;
}
*/
```

3. Implemente e teste um programa que crie uma lista encadeada com os nós de uma árvore AVL em percurso em-ordem. O programa deverá imprimir o resultado do percurso em-ordem na AVL e do conteúdo armazenado na lista encadeada.

```c
#include <stdio.h>
#include <stdlib.h>
#include "ArvoreAVL.h"
#include "Lista.h"

int main(){
    ArvAVL* avl;
    Lista* emOrdem;
    int res,i;
    int N = 10, dados[10] = {50,25,10,5,7,3,30,20,8,15};

    avl = cria_ArvAVL();
    emOrdem = lista_cria();


    for(i=0;i<N;i++){
        res = insere_ArvAVL(avl,dados[i]);
    }

    printf("\nAVL em ordem:\n");
    emOrdem_ArvAVL(avl);
    printf("\n\n");

    printf("\nLista em ordem:\n");
    emOrdem_Lista(avl, emOrdem);

    printf("%s",imprime_lista(emOrdem));
    printf("\n\n");

    libera_ArvAVL(avl);
    lista_libera(emOrdem);

    return 0;
}
```

```c
//Função que gera a lista

void emOrdem_Lista(ArvAVL *raiz, Lista* q){

    static int contador = 0;

    if(raiz == NULL)
        return;
    if(*raiz != NULL){
        emOrdem_Lista(&((*raiz)->esq), q);
        lista_insere_posicao(q,(*raiz)->info,contador);
        contador++;
        emOrdem_Lista(&((*raiz)->dir), q);
    }
}
```

Observação:

- Várias partes do código foram retiradas devido ao tamanho, mas tenho todos salvos para enviar se preciso.
- No exercício 3, a lista é exibida, contudo com erros no final. Isso acontece por que peguei um código do semestre passado de lista, e continha alguns erros na liberação (free) da lista.