Centro Universitário Farias Brito	**CURSO:** CIÊNCIA DA COMPUTAÇÃO **DISCIPLINA:** ESTRUTURAS DE DADOS II **PROFESSOR(A): VITOR ALMEIDA DOS SANTOS**	**ATIVIDADES DA AULA DE ÁRVORES (IMPLEMENTAÇÕES)**
	Aluno(a):_____	

Atividades do conteúdo: Árvores (Implementação)

3-1 Desenvolva o método de pós-ordem de uma árvore ternária.

arvore.h

```
typedef struct NoT TNoT;

struct NoT{

        int valor;
        TNoT *esq;
        TNoT *dir;
        TNoT *meio;

};

TNoT *criarT(int v);
void posordemT(TNoT *r);
```

arvore.c

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include "arvore.h"

TNoT *criarT(int v){

        TNoT *r = (TNoT *)malloc(sizeof(TNoT));
        r->valor = v;
        r->esq = NULL;
        r->dir = NULL;
        r->meio = NULL;

        return r;
}
```

```
void posordemT(TNoT *r){

	if(r->esq != NULL){

		posordemT(r->esq);

	}

	if(r->meio != NULL){

		posordemT(r->meio);

	}

	if(r->dir != NULL){

		posordemT(r->dir);

	}

	printf("%d\n", r->valor);
}
```

3 - 2 Escreva um método que receba a raiz de uma árvore binária e retorne a soma de todos os seus valores ímpares.

```
arvore.h

typedef struct No TNo;

struct No{

	int valor;
	TNo *esq;
	TNo *dir;

};

TNo *criar(int v);
int somaimpar(TNo *r);
```

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include "arvore.h"
```

arvore.c

```
int somaimpar(TNo *r){

	int esq = 0, dir = 0;

	if(r->esq != NULL){

		esq = somaimpar(r->esq);

	}

	if(r->dir != NULL){

		dir = somaimpar(r->dir);

	}

	if(r->valor % 2 != 0){

		return esq + dir + r->valor;

	} else{

		return esq + dir;

	}
}
```

3 - 3 Uma árvore binária pode ter elementos repetidos. Escreva um método que receba a raiz de uma árvore binária e um valor de elemento e retorne a quantidade de ocorrências deste valor na árvore.

Arvore.h

```
typedef struct No TNo;

struct No{

	int valor;
	TNo *esq;
	TNo *dir;

};
```

```
int numrep(TNo *r, int n);
```

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include "arvore.h"

Arvore.c

int numrep(TNo *r, int n){

	int esq = 0, dir = 0;

	if(r->esq != NULL){

		esq = numrep(r->esq, n);

	}

	if(r->dir != NULL){

		dir = numrep(r->dir, n);

	}

	if(r->valor == n){

		return esq + dir + 1;

	} else{

		return esq + dir;

	}
}
```

3 - 4 Escreva um método que receba a raiz de uma árvore n-ária e retorne a soma de todos os seus valores.

```
Arvore.h

typedef struct No TNo;

struct No{

	int valor;
```

```
        TNo *esq;
        TNo *dir;

};

void somavalor(TNoN *r, int n);
```

Arvore.c

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include "arvoren.h"

void somavalor(TNoN *r, int n){

        int i, soma = 0;

        r->valor += n;

        for(i = 0; i < r->qtd; i++){

                if(r->filhos[i] != NULL){

                        soma = somavalor(r->filhos[i], n);

                }
        }
        return soma + r->valor;
}
```

3 - 5 Escreva um método que receba a raiz de uma árvore n-ária e um valor n e adicione esse valor a todas os nós da árvore.

```
typedef struct NoN{

        int valor;
        int qtd;
        struct NoN **filhos;

} TNoN;

void inserevalor(TNoN *r, int n);
```

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include "arvoren.h"

arvoren.c

void inserevalor(TNoN *r, int n){

        int i;

        r->valor += n;

        for(i = 0; i < r->qtd; i++){

                if(r->filhos[i] != NULL){

                        inserevalor(r->filhos, n);

                }

        }

}
```

3 - 6 Escreva um método que receba a raiz de uma árvore binária e um valor n e insira um nó com esse valor como filho da folha mais a direita.

```
arvore.h

typedef struct No TNo;

struct No{

        int valor;
        TNo *esq;
        TNo *dir;

};

void inserirdir(TNo *r, int n);
```

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include "arvore.h"

arvore.c

void inserirdir(TNo *r, int n){

        TNo *ptno = r;

        while(ptno->dir != NULL){

                ptno = ptno->dir;

        }
        ptno->dir = criar(n);
}
```

3 - 7 Escreva um método que receba as raízes de duas árvores binárias e retorne 1 se elas forem iguais e 0 se forem diferentes.

```
arvore.h

typedef struct No TNo;

struct No{

        int valor;
        TNo *esq;
        TNo *dir;

};

int comparar(TNo *a, TNo *b);
```

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include "arvore.h"

arvore.c


int comparar(TNo *a, TNo *b){
```

```
        int resultdir = 0, resultesq = 0;

        if (a->valor != b->valor){

                return 0;

        }else {

        resultdir = comparar(a->dir, b->dir);
        resultesq = comparar(a->esq, b->esq);
        return resultdir * resultesq;

        }
}
```