Aluno: Luiz Rodrigo Lacé Rodrigues

DRE: 118049873

**Questão 1)**

//struct para os nós da lista encadeada

typedef struct tno{

int valor;

struct tno \*prox;

} tno;

void insere(tno \*\*lista, int x){

tno \*novo;

tno \*aux;

//aloca memoria para o novo

novo = (tno \*) malloc(sizeof(tno));

//nó novo recebe o valor passado pelo parâmetro

novo->valor = x;

//para salvar o valor anterior, mas não é uma lista duplamente encadeada, como visto na struct

tno \*anterior;

// faz o auxiliar ser igual a lista, anterior por enquanto não recebe nada

aux = \*lista;

anterior = NULL;

//Caso a lista comece vazia o nosso valor inserido será o inicio dela

if(aux == NULL){

novo->prox = NULL;

\*lista = novo;

}

// caso a lista já tenha elementos

else{

// utilizamos o auxiliar para, caso seja diferente de nulo e menor que o valor x que queremos inserir, vamos percorrer para o próximo salvando sempre o anterior.

while(aux != NULL && aux->valor < x){

anterior = aux;

aux = aux->prox;

}

// quando encontrarmos algum valor que seja maior ou nulo, fazemos o proximo ser o novo

novo->prox = aux;

// caso o anterior salvo seja nulo,fazemos a lista ser igual ao novo nó

if(anterior == NULL){

\*lista = novo;

}

// caso contrario, o proximo do anterior vai ser o nosso nó que estamos inserindo

else{

anterior->prox = novo;

}

}

}

**Questão 2)**

**2.1)**

tno \* soma ( tno \* ptraiz){

// caso a raiz seja nula, vamos retornar 0. É importante para a chamada recursiva

if(ptraiz == NULL){

return 0

}

// a raiz passada como parâmetro vai receber no seu campo soma o valor da sua chave mais a chamada recursiva tanto dos nós à direita quanto esquerda

ptraiz-> soma = raiz-> chave + soma(ptraiz -> esq) + soma(ptraiz->dir);

// retornamos a chave para que a recursividade seja feita com sucesso

return ptraiz->chave;

}

**2.2)**

void insere(tno \* ptraiz, int x){

// inserção recursiva, caso raiz seja nula, criamos um novo nó

if(ptraiz == NULL){

tno \* novo = (tno\*) malloc(sizeof(tno)));

novo->esq = NULL;

novo->dir = NULL;

novo->chave = x;

novo->soma = x;

return novo;

}

else{

if(x < ptraiz->chave){

// adicionando o valor x na soma a cada nó que for do “caminho”

ptraiz->soma += x;

// chamada recursiva

ptraiz->esq = insere(ptraiz->esq, x);

}

if(x > ptraiz->chave){

// adicionando o valor x na soma a cada nó que for do “caminho”

ptraiz->soma += x;

// chamada recursiva

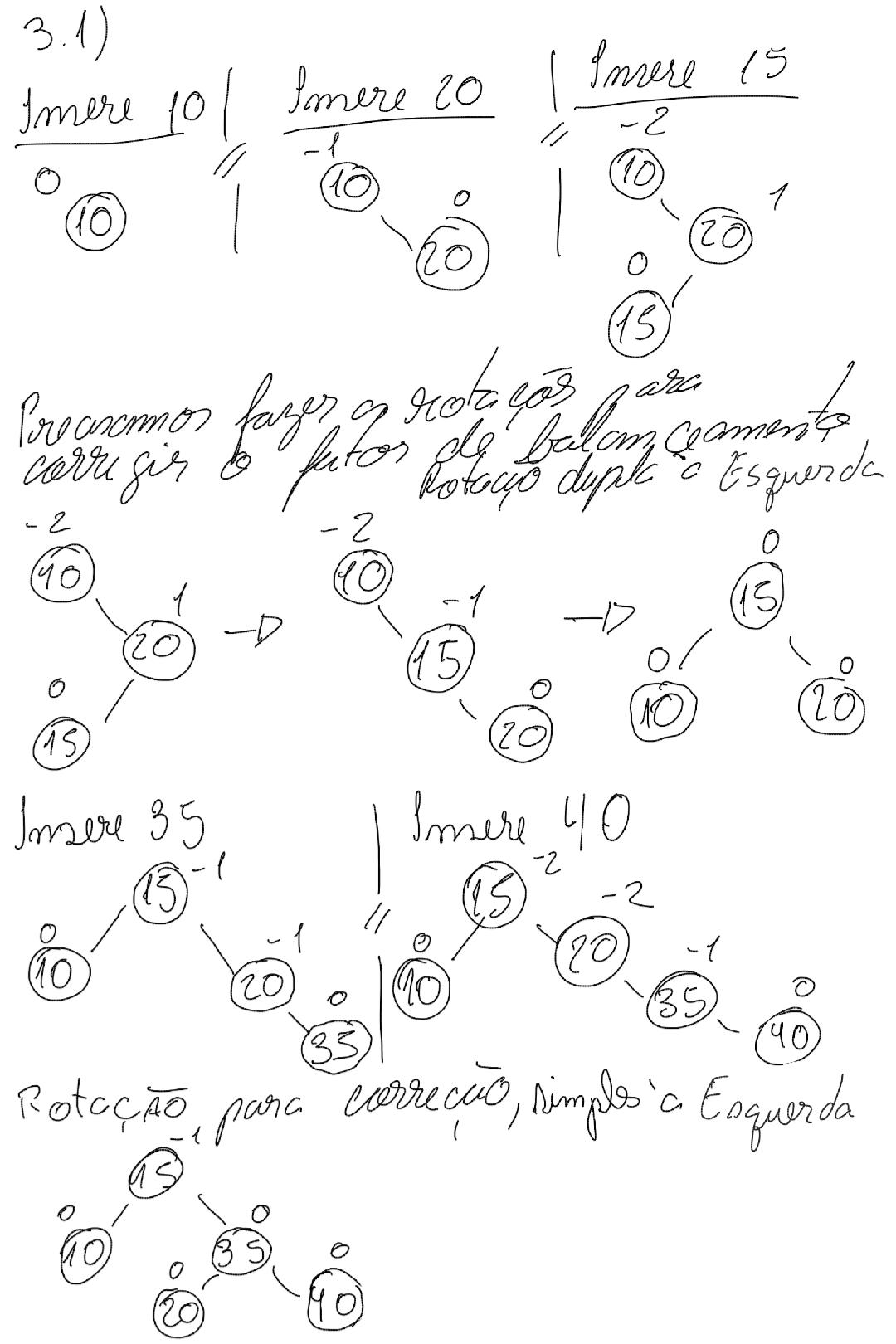
ptraiz->dir = insere(ptraiz->dir, x);

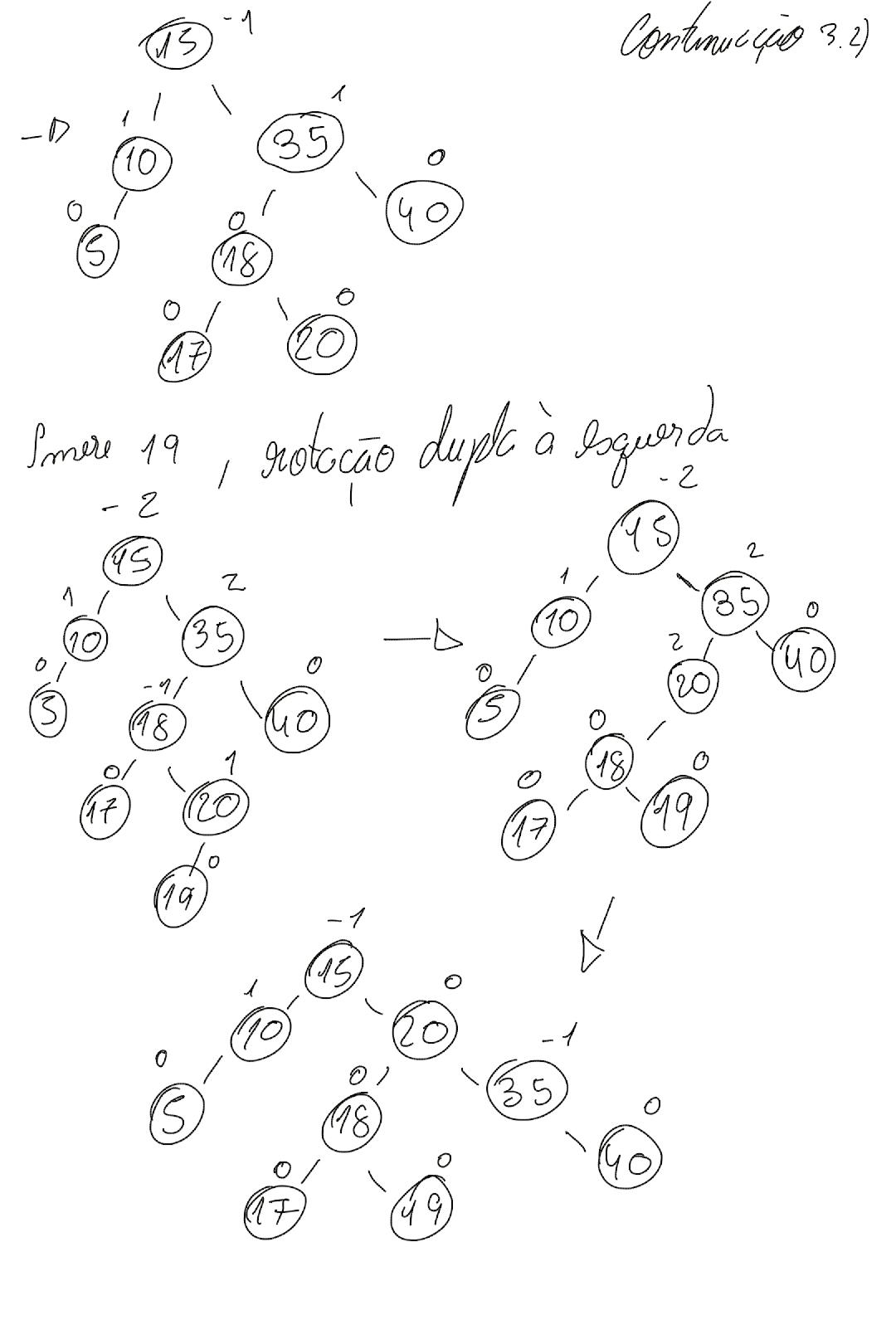
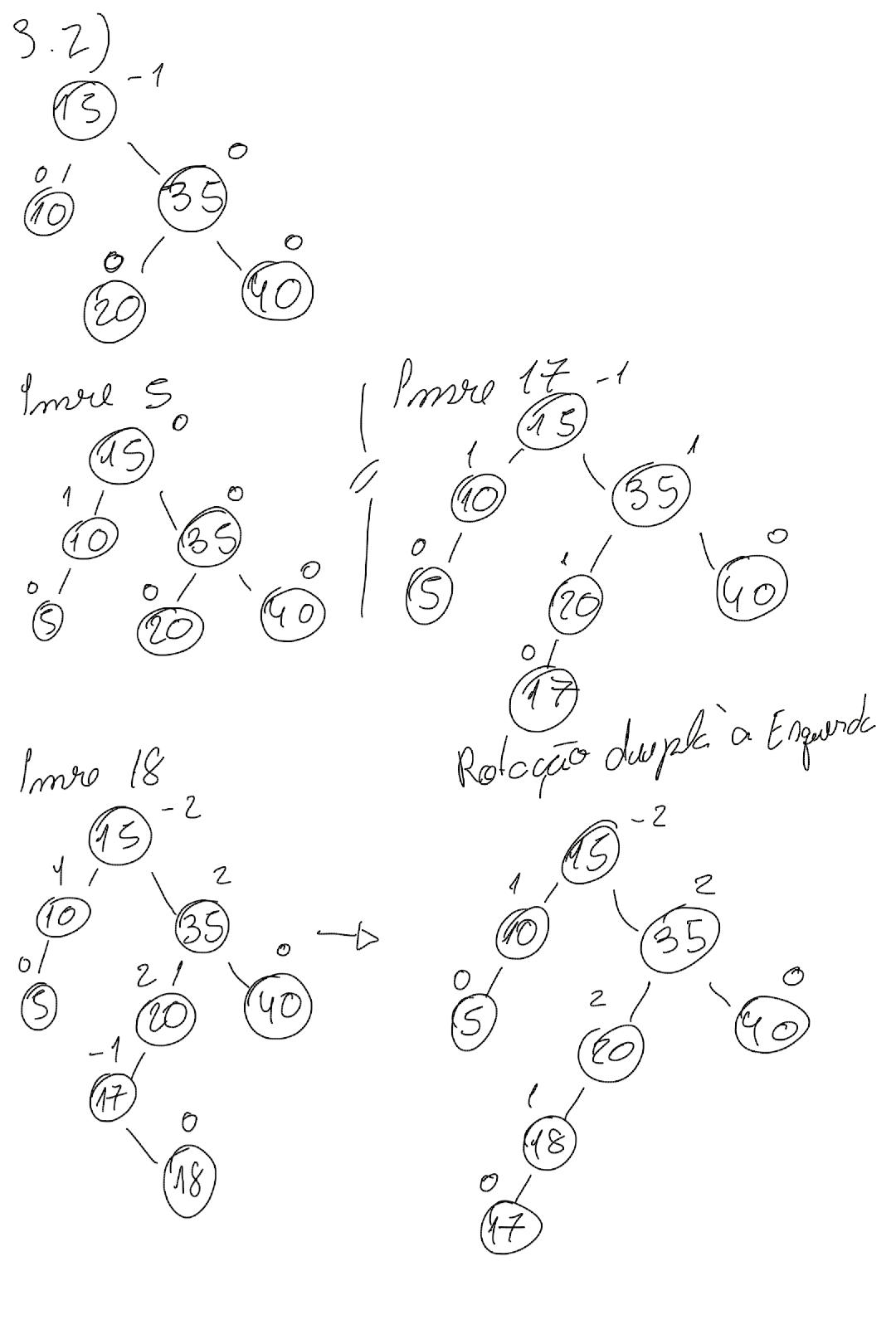
}

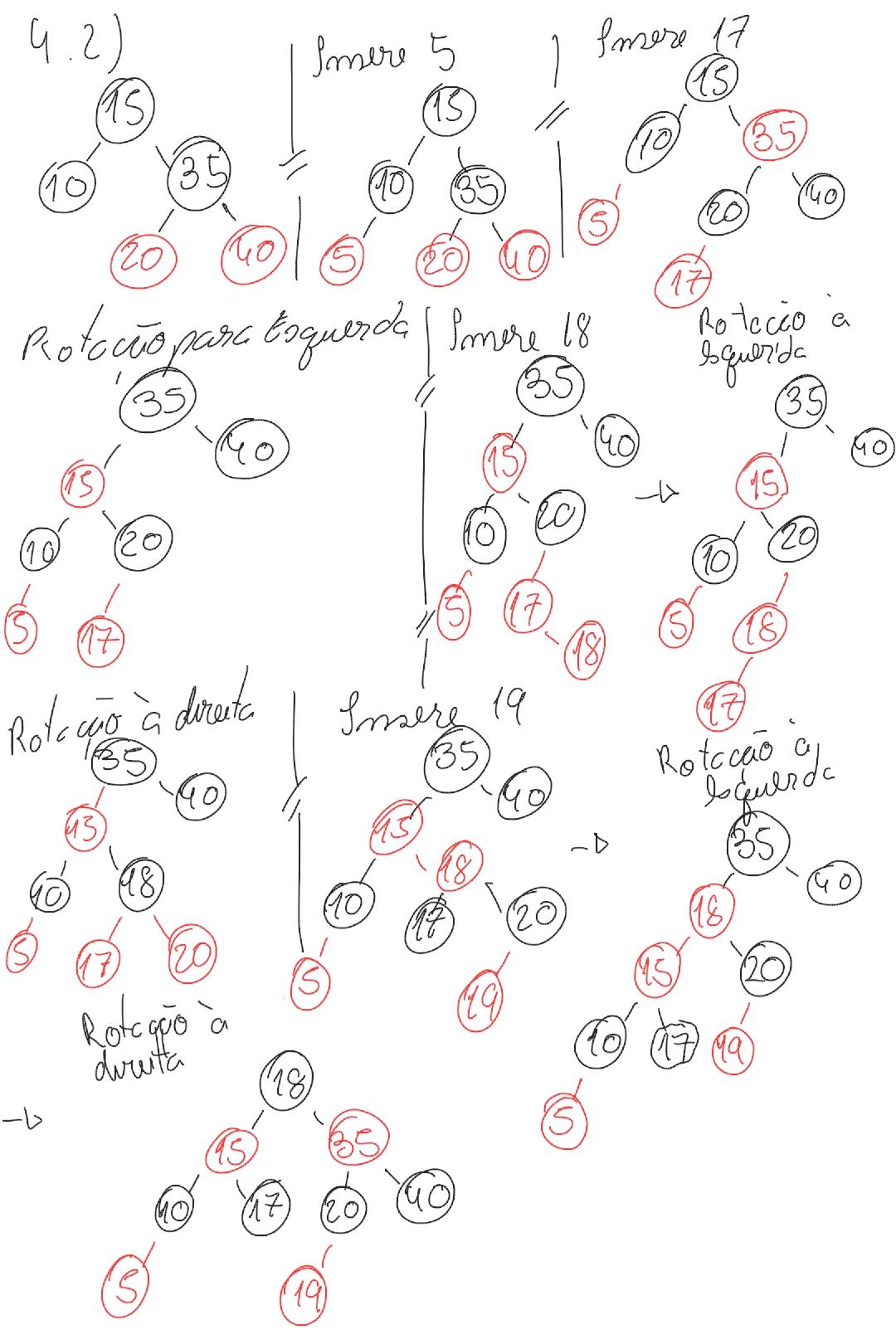
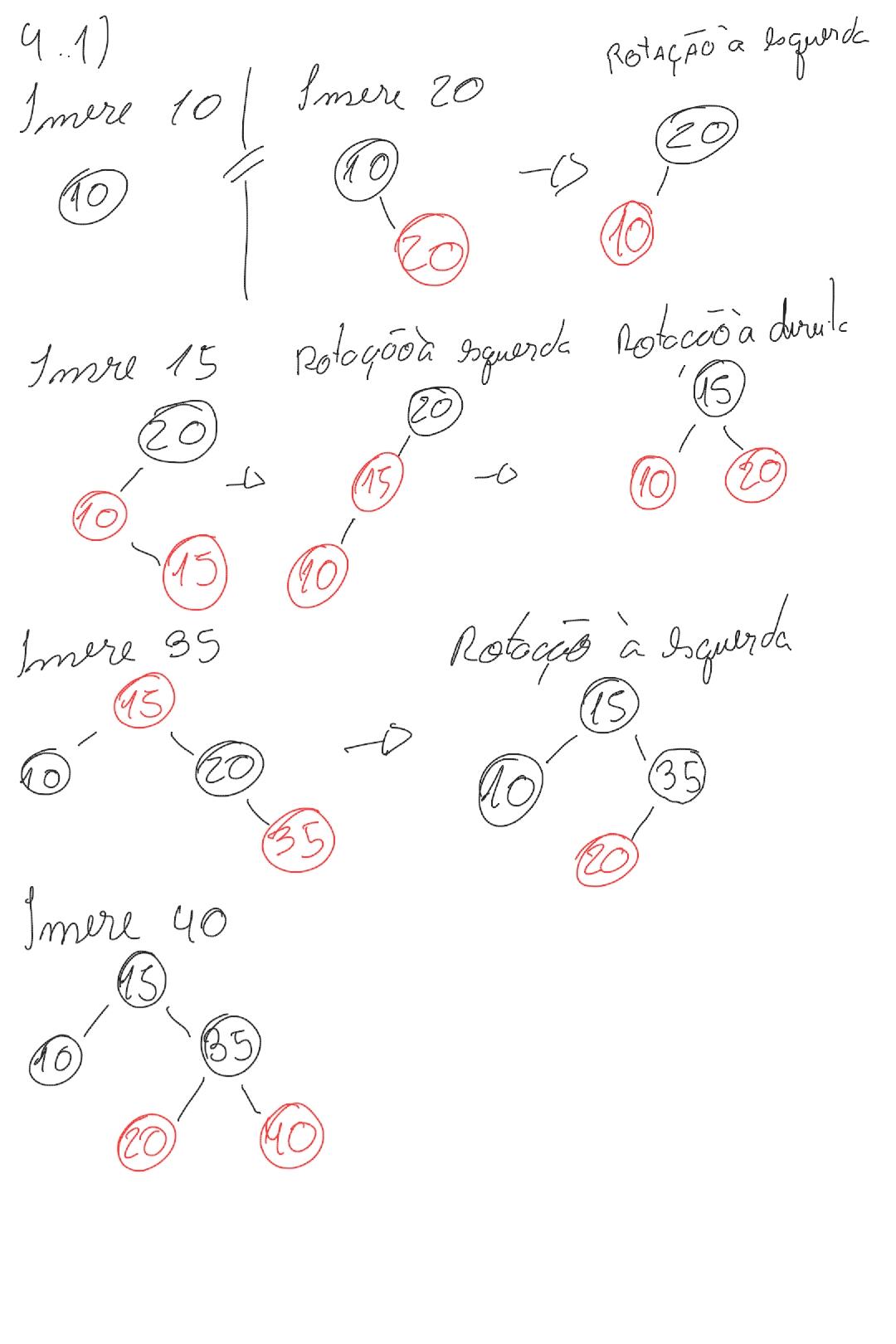
return ptraiz;

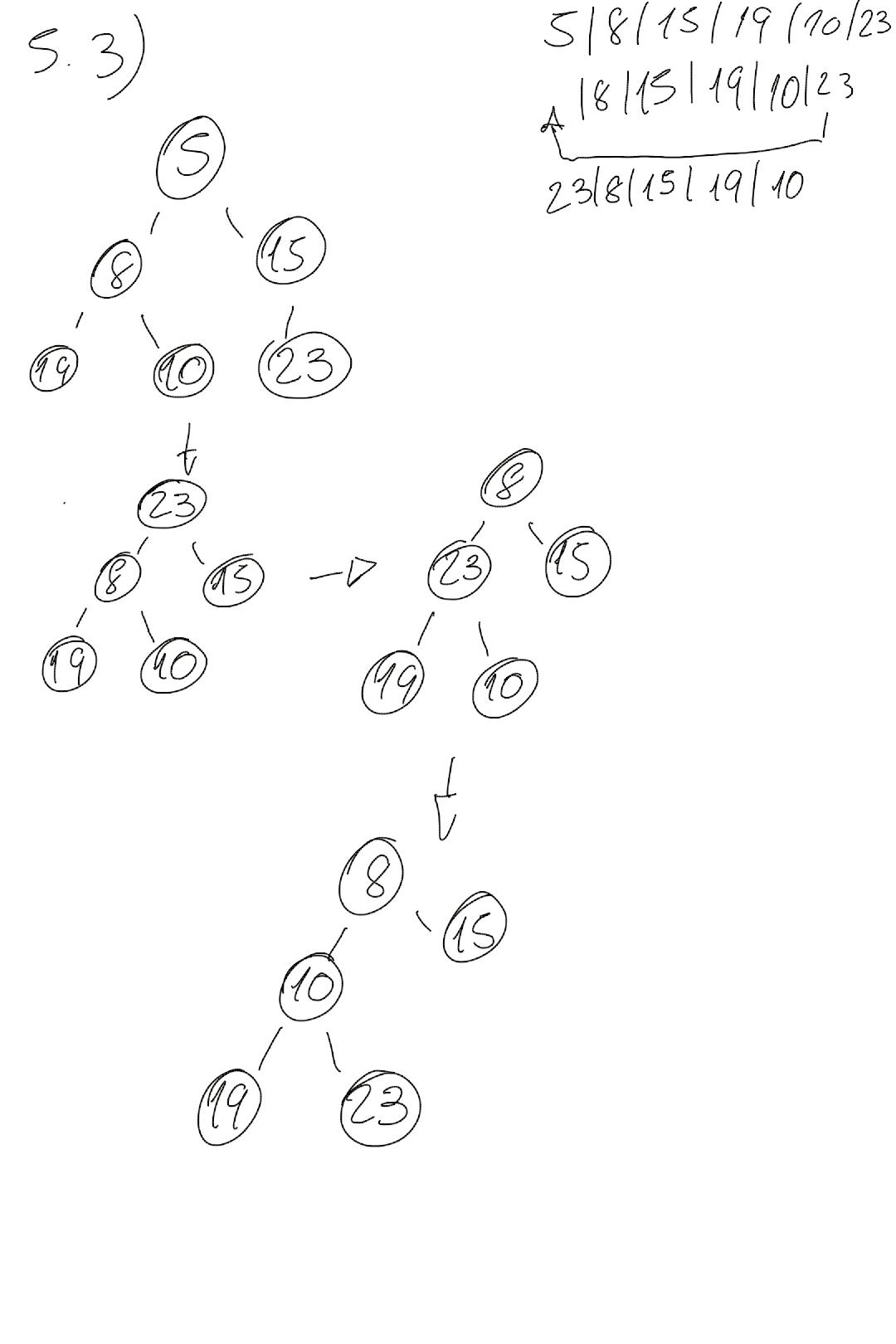
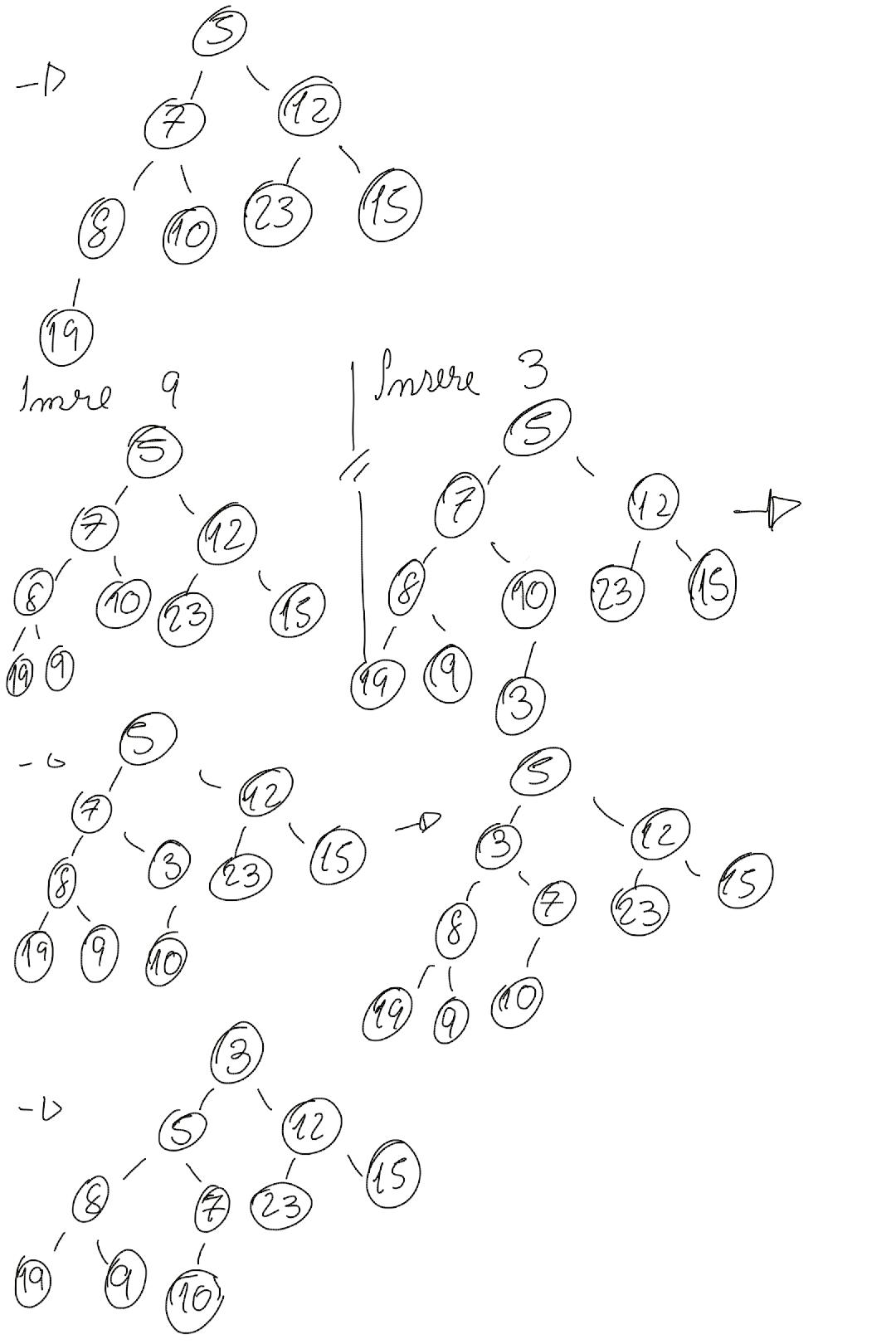
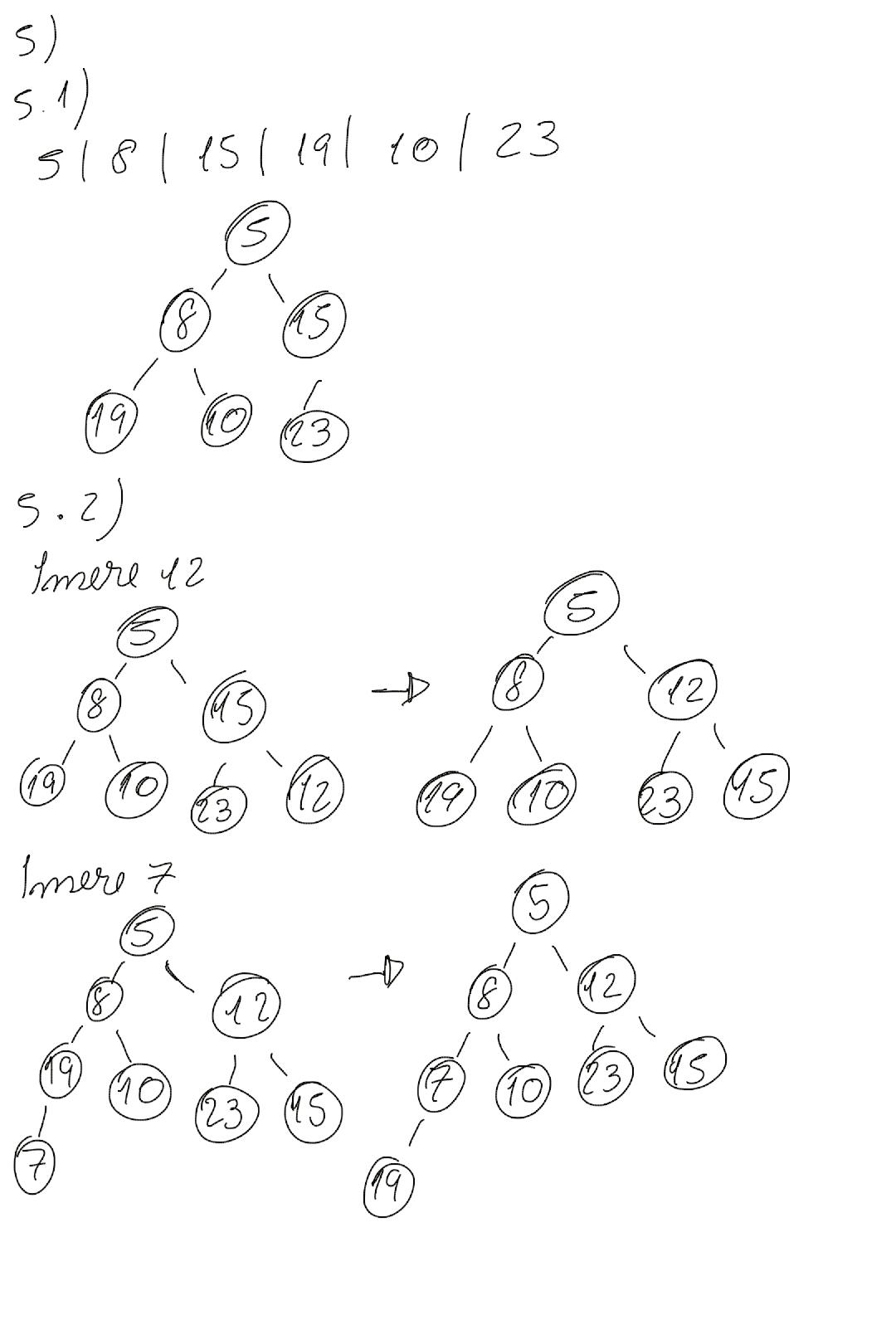
}

}

**Questão 3)**



**Questão 4)**

**Questão 5)**