Estrutura de Dados, 2021.1 - Prova 1 - Resoluções

David Rodrigues Albuquerque - 120047390 - davidra@dcc.ufrj.br

Outubro, 2021

Questão 1) Para a questão 1), foi usada as seguintes definições e funções.

```
typedef struct Node {
    int value;
    struct Node *next;
} Node;
```

Para inserir o elemento numa lista encadeada sem nó cabeça de forma ordenada, foi utilizado o seguinte algoritmo

```
Node* insert_ordered_node(Node *list, int new_node_value) {

    Node *aux, *ant;

    /**
    * lista vazia
    **/
    if(list == NULL) {
        list = (Node *)malloc(sizeof(Node));
        list->value = new_node_value;
        list->next = NULL;
        return list;
    }

    aux = list;

    /**
    * Valor deve ser inserido no começo da lista
    **/
    if(new_node_value < list->value) {
        list = (Node *)malloc(sizeof(Node));
        list->value = new_node_value;
        list->next = aux;
        return list;
    }
```

```
    while(new_node_value > aux->value && aux->next != NULL) {
        ant = aux;
        aux = aux->next;
    }

    /**
    * Valor deve ser inserido no final da lista
    **/
    if(new_node_value > aux->value) {
        aux->next = (Node *)malloc(sizeof(Node));
        aux = aux->next;
        aux->value = new_node_value;
        aux->next = NULL;
        return list;
    }

    /**
    * Valor deve ser inserido no meio da lista
    **/
    ant->next=(Node *)malloc(sizeof(Node));
    ant=ant->next;
    ant->value=new_node_value;
    ant->next=aux;
    aux=NULL;
    ant=aux;

    free(aux);
    free(ant);

    return list;

}
```

Questão 2) Para ambas as questões 2.1) e 2.2), foi usada as seguintes definições e funções. A função calcsum é responsável por, dado um nó, calcular a sua soma, buscando recursivamente o valor das somas a esquerda e direta. Se estas ainda não foram calculadas, a soma é chamada para esse nó filho também. Do contrário, o valor já é aproveitado, economizando processamento.

```
typedef struct Node {
    int chave;
    struct Node *esq;
    struct Node *dir;
    int soma;
} Node;

int calc_sum(Node* node) {

    int sum = 0;
    if(node != NULL) {
        if(node->soma != 0) {
            return node->soma;
        }
        sum += calc_sum(node->esq);
        sum += calc_sum(node->dir);
        node->soma = node->chave + sum;
        return node->soma;
    }
    else {
        return 0;
    }
}
```

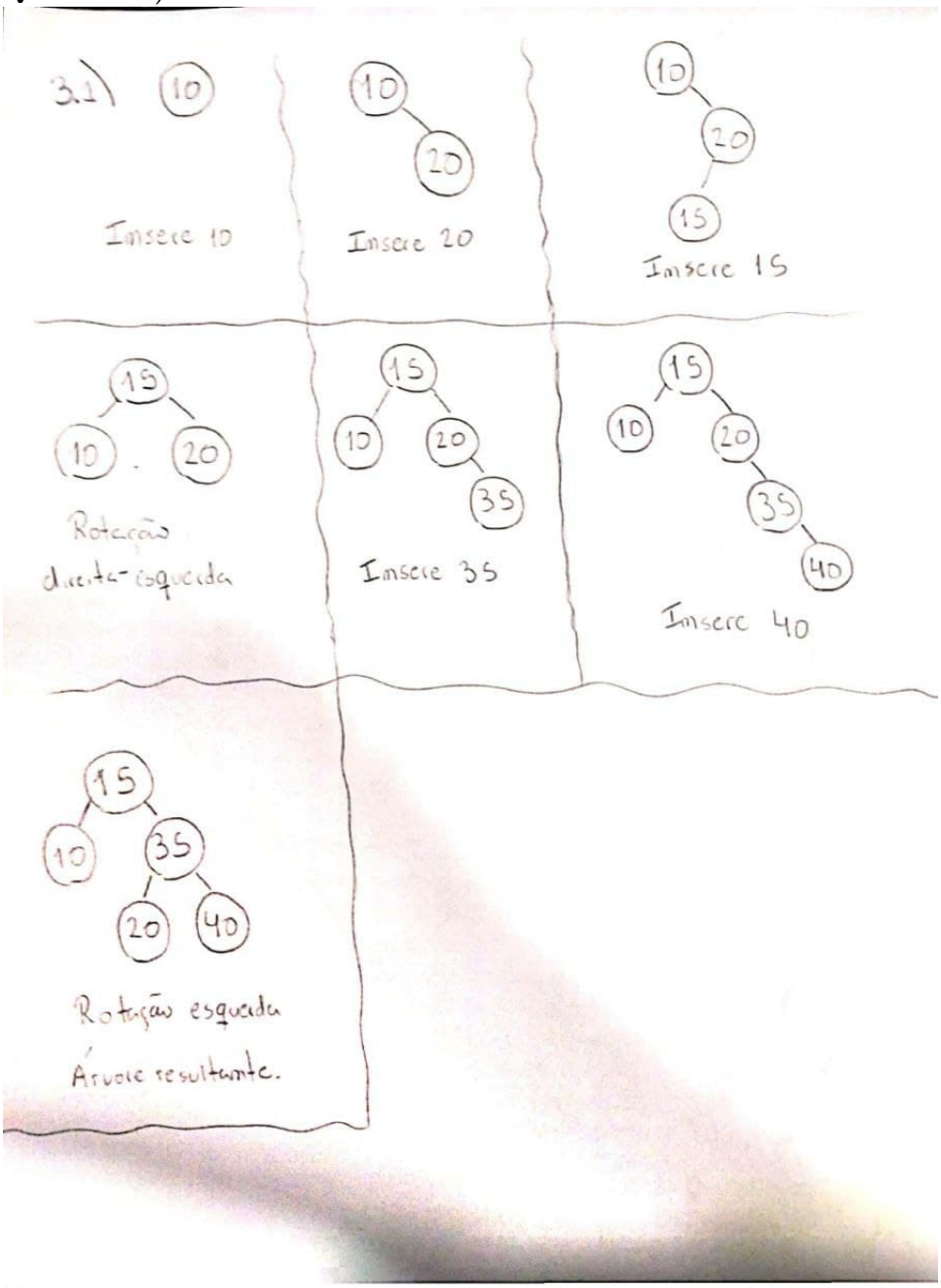
Questão 2.1) Este algoritmo preenche a a soma de um nó dado e de suas sub-árvores. Desta forma, basta apenas passar o nó raiz que calcularemos toda a árvore, pois a função calcsum, armazena também os valores das somas dos nós filhos, se ainda não estiverem calculados.

```
Node* fill_all_sums(Node* root) {

    if(root == NULL) {
        return NULL;
    }

    root->soma = calc_sum(root->esq) + calc_sum(root->dir) + root->chave;
    return root;
}
```

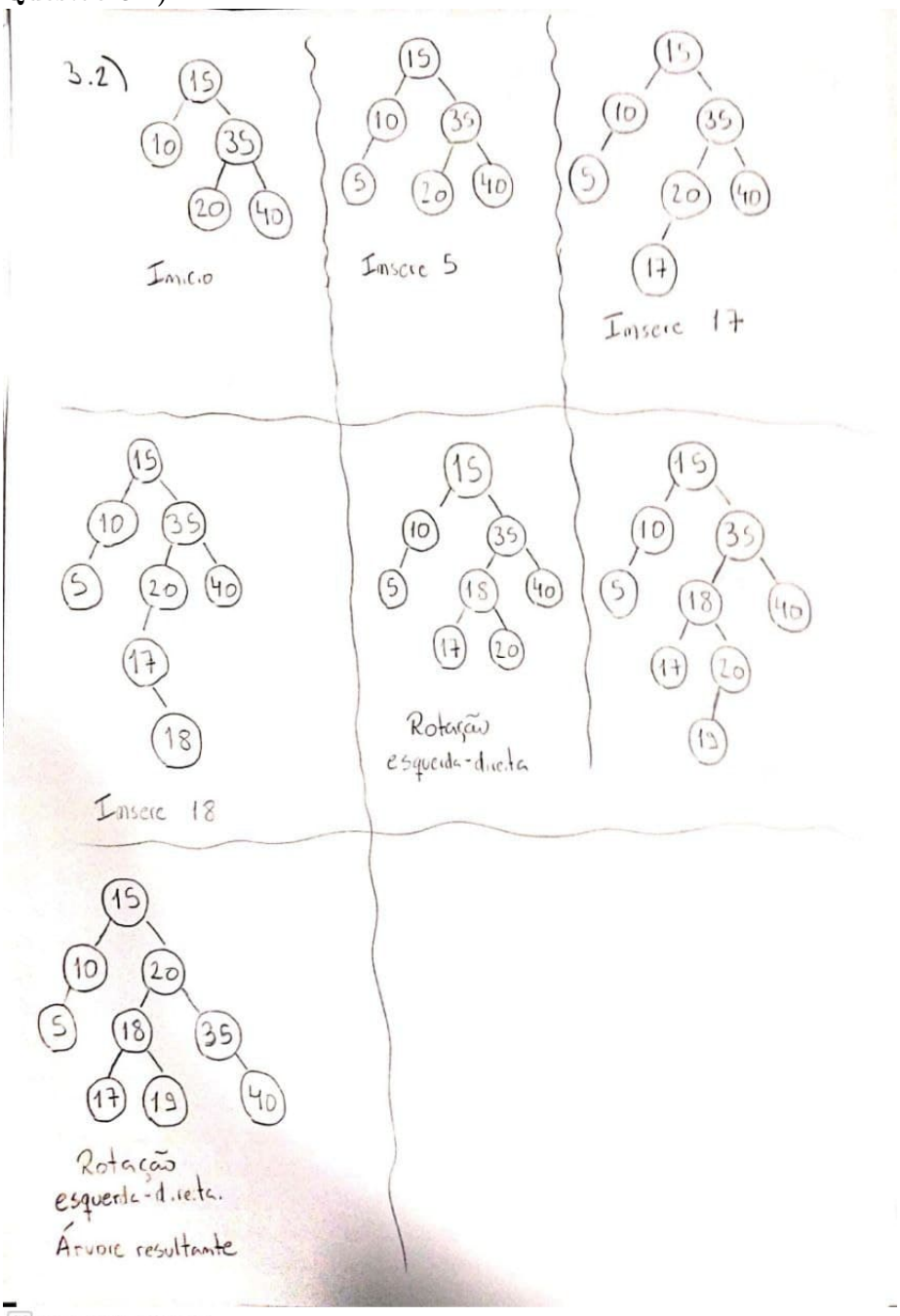
Questão 2.2) Este algoritmo insere um elemento na árvore e recalcula o valor das somas dos nós afetados.

```
Node* insert_ordering(int value, Node *root) {

    /**
    * Se árvore vazia.
    **/
    if( root == NULL ) {
        return add_node(value);
    }

    /**
    * Se valor repetido.
    **/
    if( value == root->chave ) {
        return root;
    }

    /**
    * Inserir esquerda e calcular soma
    **/
    if( value < root->chave ) {
        root->esq = insert_ordering(value,root->esq);
        root->soma = calc_sum(root->esq) + calc_sum(root->dir) + root->chave;
    }

    /**
    * Inserir direita e calcular soma
    **/
    if( value > root->chave ) {
        root->dir = insert_ordering(value,root->dir);
        root->soma = calc_sum(root->esq) + calc_sum(root->dir) + root->chave;
    }

    return root;
}
```

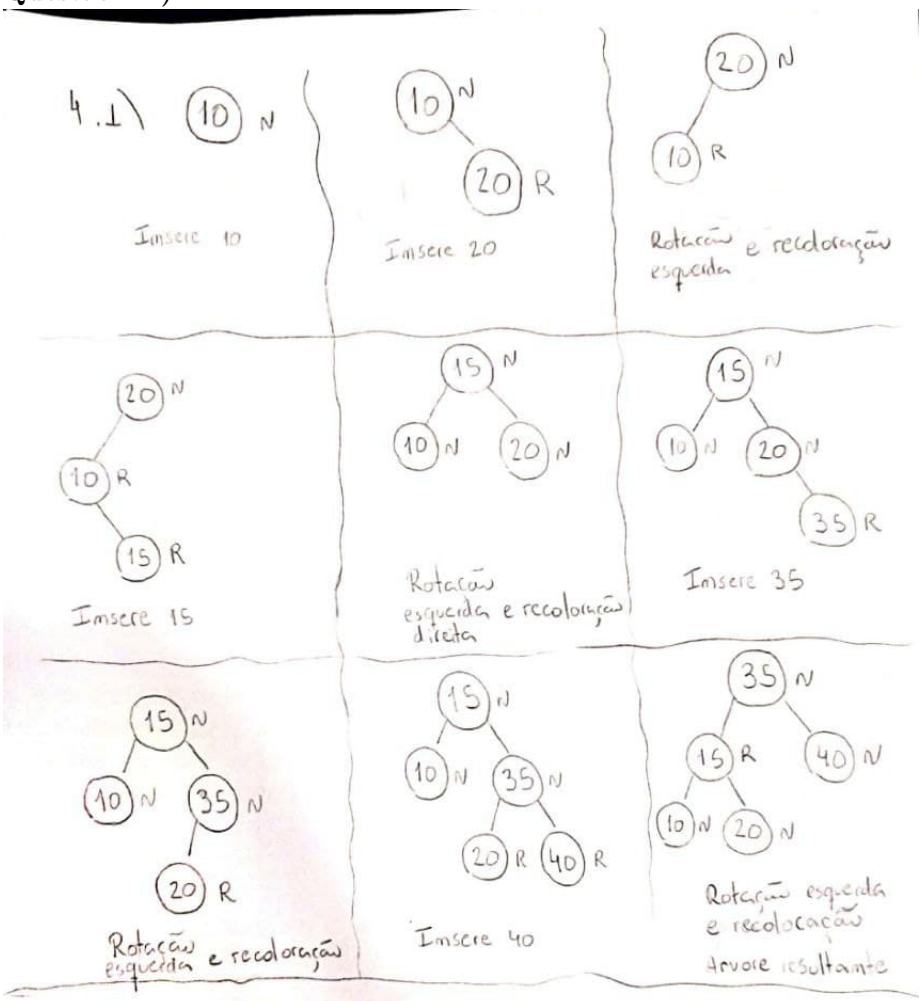
Questão 3.1)



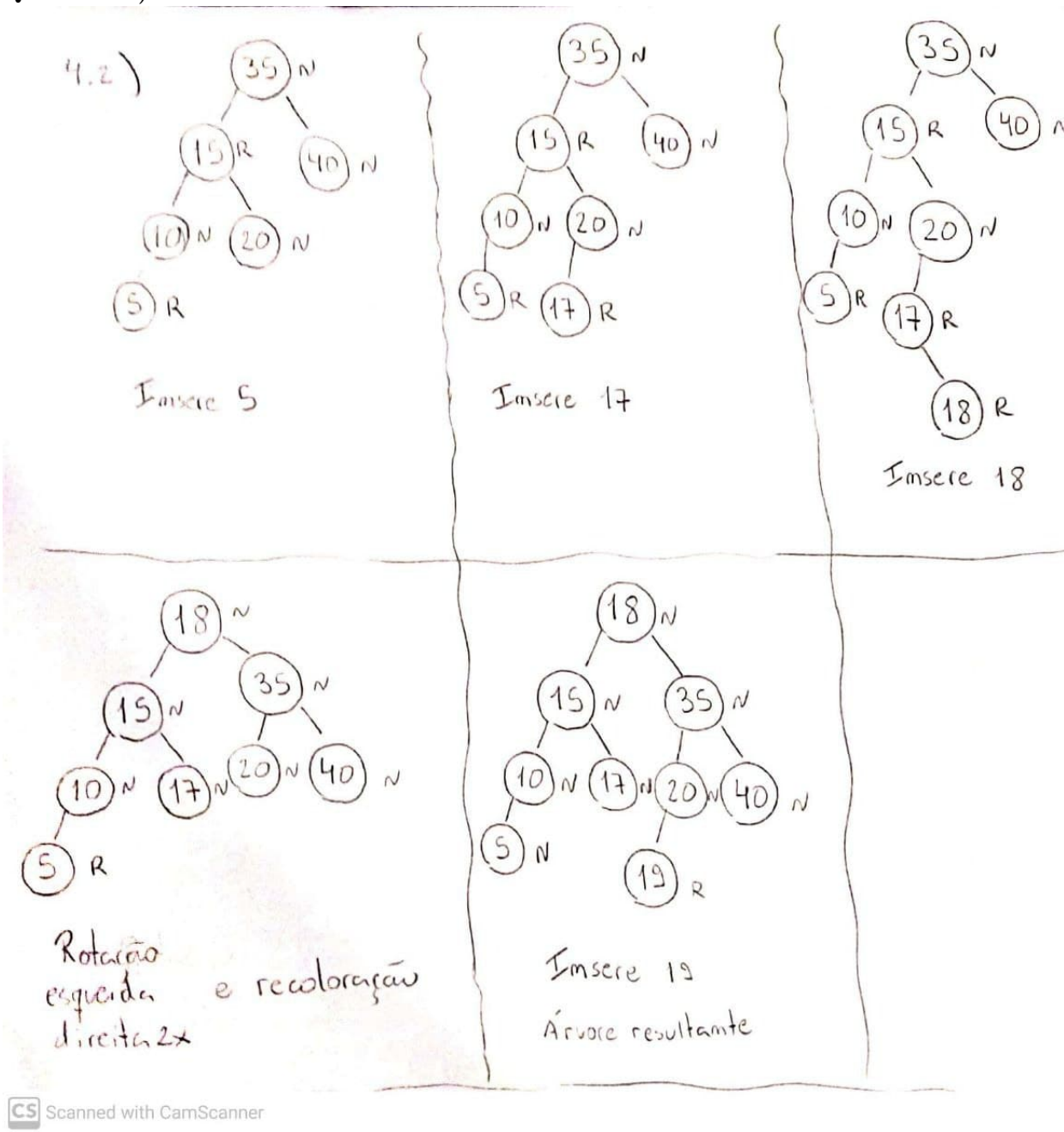
Questão 3.2)


3.2)
Início
Insere 5
Insere 17
Insere 18
Rotação esquerda-direita
Rotação esquerda-direita.
Árvore resultante
Scanned with CamScanner

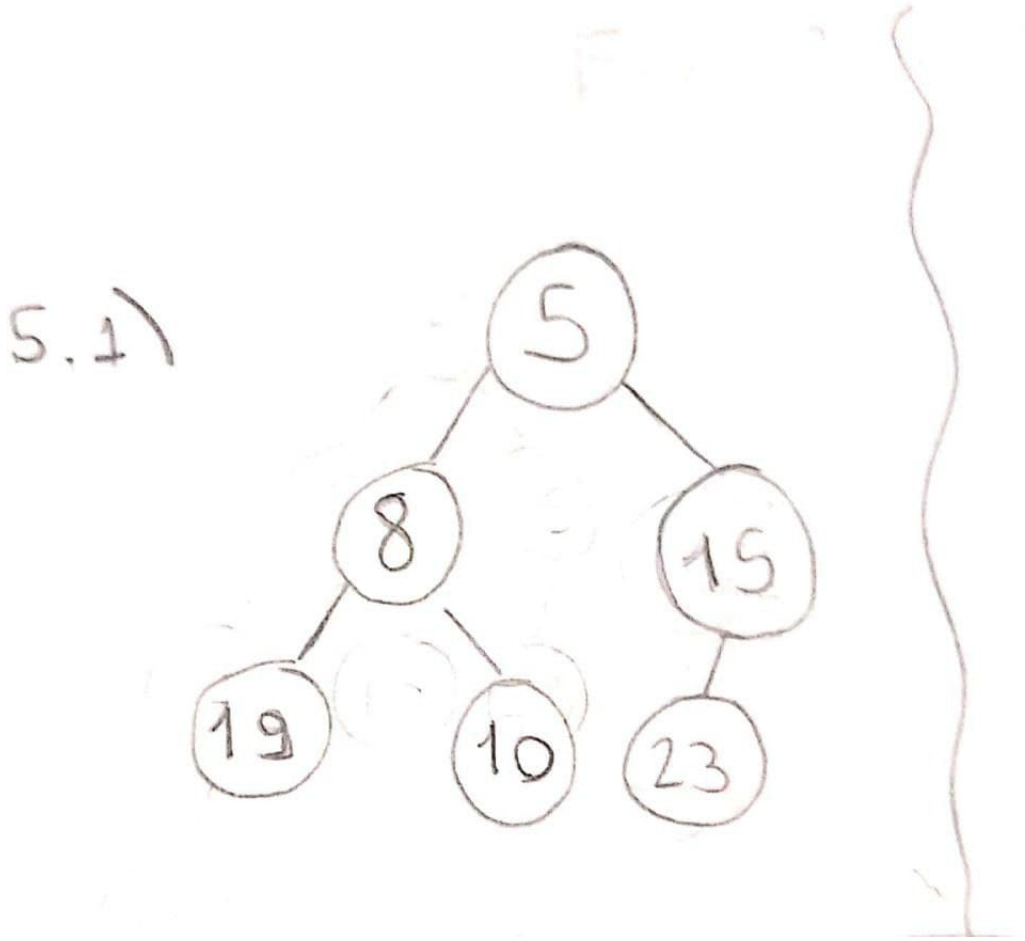
Questão 4.1)



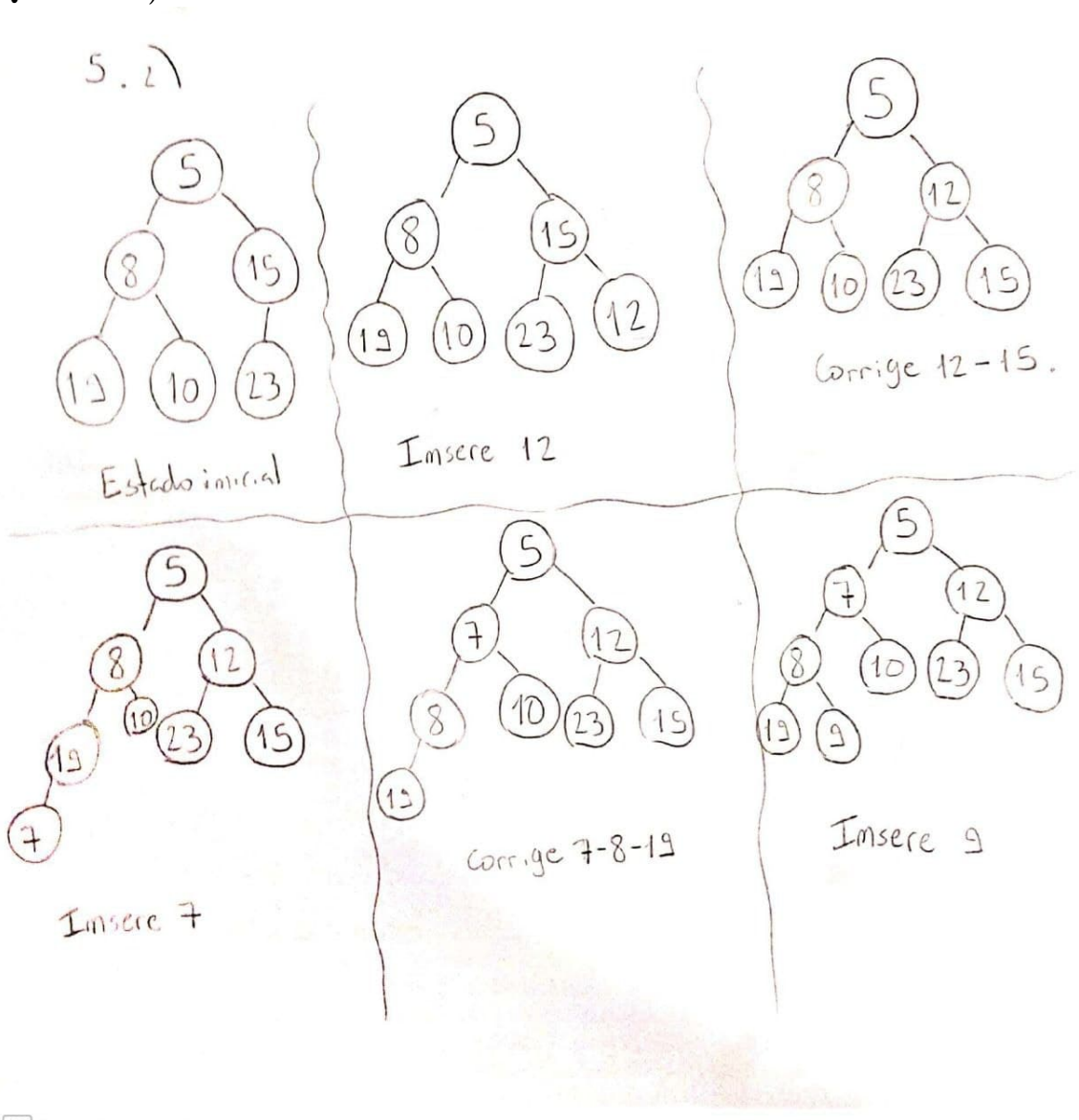
Questão 4.2)

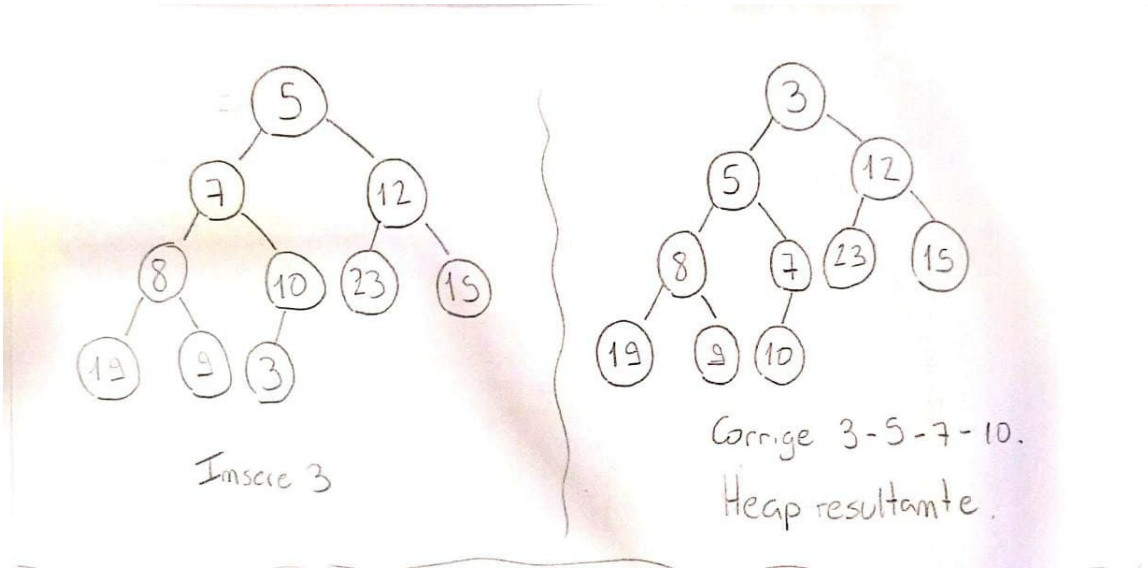


Questão 5.1)

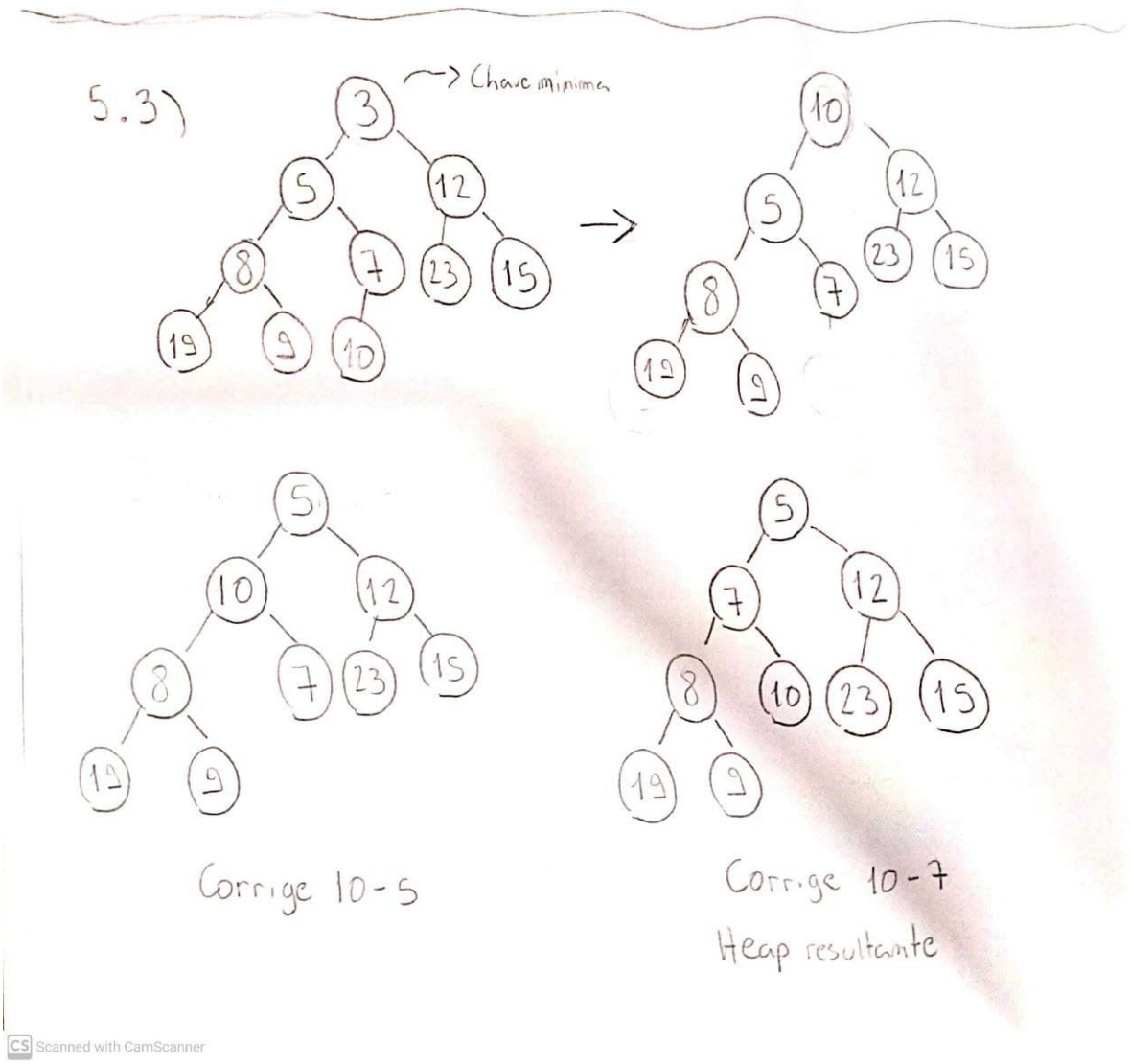


Questão 5.2)





Questão 5.3)


5.3)
Chave mínima
Corrige 10-5
Corrige 10-7
Heap resultante
Scanned with CamScanner