**Exercício 1**

**Lista Ligada**

Estrutura de dados linear que utiliza alocação dinâmica de memória. Os elementos indicam qual é o seguinte por meio de ponteiros – endereço de memória de outra variável. Desse modo, não é necessário que sua ordem coincida com sua posição física em memória.

Operações:

* Inserir um elemento no final da lista
* Inserir um elemento no início da lista
* Inserir um elemento em qualquer posição da lista
* Pesquisar um elemento
* Remover um elemento no final da lista
* Remover um elemento no início da lista
* Remover um elemento em qualquer posição da lista

**Pilha**

São listas que utilizam a política LIFO (Last-In First-Out), em que o último elemento a ser inserido, será o primeiro a ser retirado. A implementação de pilhas pode ser realizada através de vetor ou através de listas encadeadas.

Operações:

* Adicionar elemento (no final)
* Remover elemento (no final)
* Verificar se a pilha está vazia
* Verificar se a pilha está cheia

**Fila de Prioridades**

São listas que utlizam a política FIFO (First-In First-Out), em que o primeiro elemento a ser inserido, será o primeiro a ser retirado. A implementação de pilhas pode ser realizada através de vetor ou através de listas encadeadas.

Operações:

* Adicionar elemento (no final)
* Remover elemento (no início)
* Verificar se a fila está vazia
* Verificar se a fila está cheia

**Fila Circular**

Um tipo especial de fila onde existe uma ligação entre o primeiro e o último item, formando assim um ciclo

Operações:

* Os mesmos de uma fila normal

**Lista Duplamente Encadeada**

A lista duplamente encadeada, difere da simplesmente encadeada apenas por possuir um atributo a mais sendo ele o “anterior”, ou seja, cada nó tem um ponteiro apontando para o seu anterior e para o seu próximo.

Operações:

* Inserir um elemento no início
* Inserir um elemento no final
* Realizar a pesquisa de um elemento
* Remover um elemento do início da lista
* Remover um elemento do final da lista

**Árvore Binária**

Uma árvore binária é uma estrutura de dados não linear, bidimensional, com propriedades especiais. Árvores são utilizadas para realizar a representação dos elementos de um determinado conjunto de dados de maneira hierárquica e está diretamente ligada à recursão.

Operações:

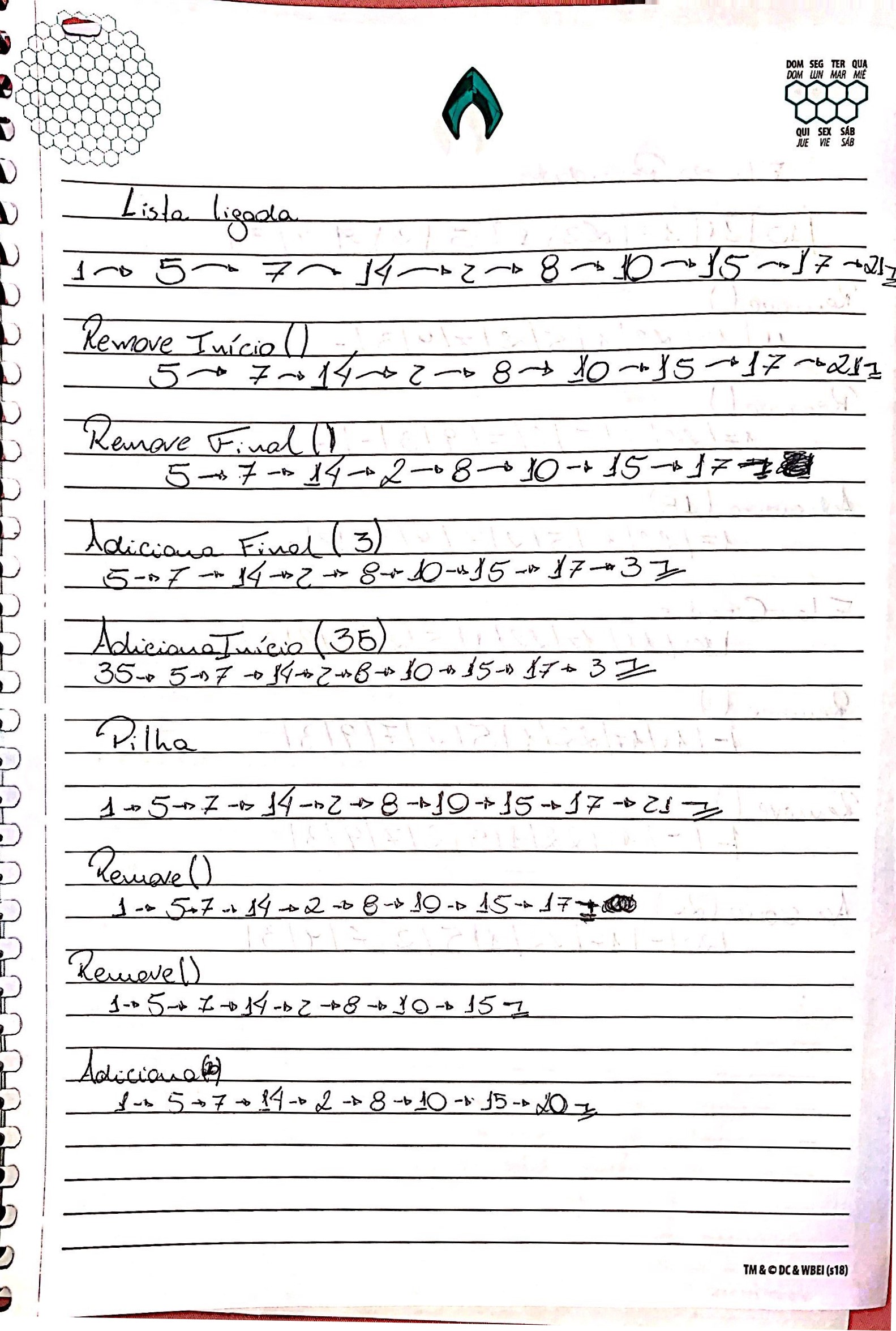
* Inserir um elemento
* Remover um elemento
* Percorrer a árvore em pré ordem
* Percorrer a árvore em ordem
* Percorrer a árvore em pós ordem

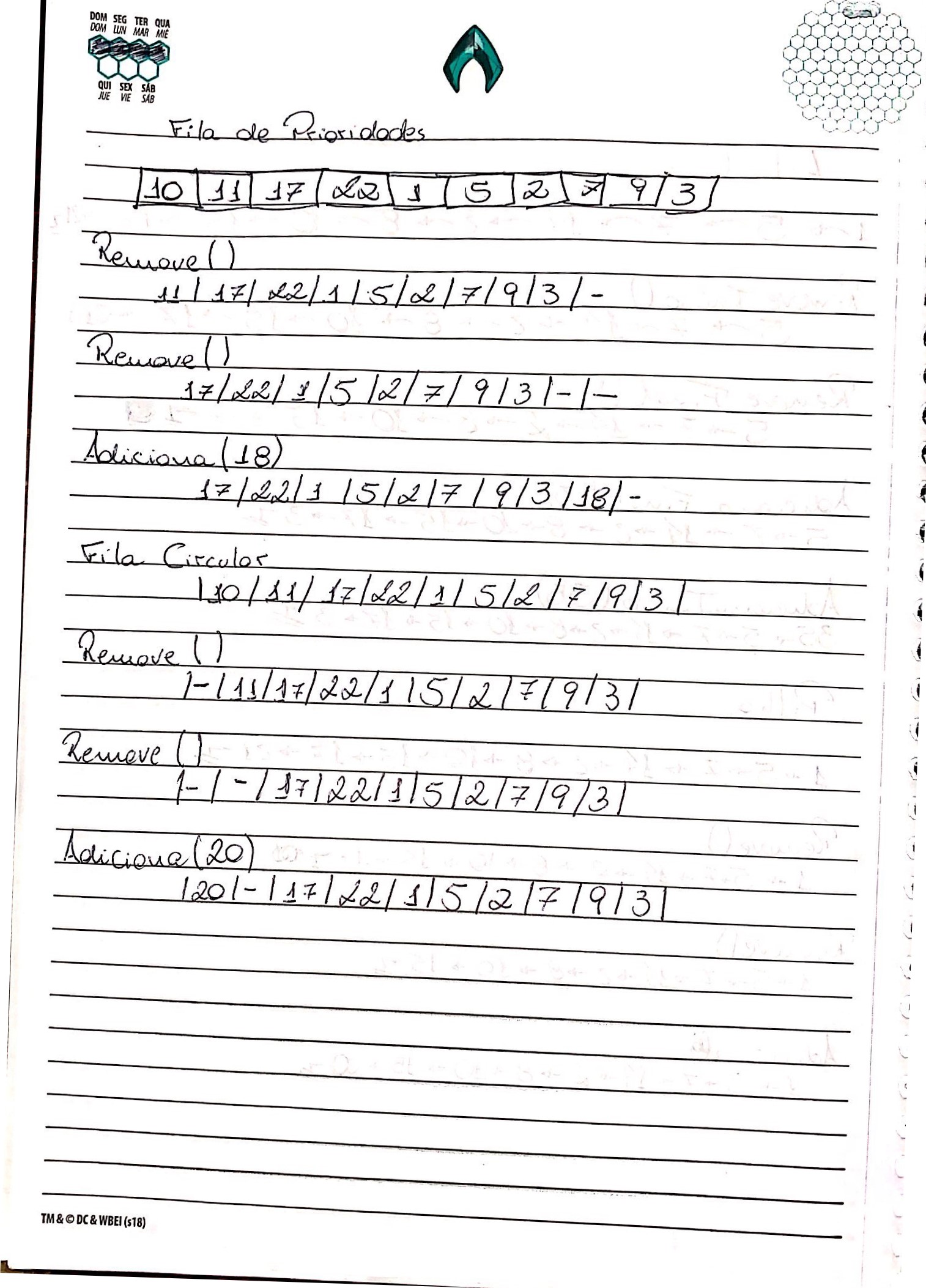
**Grafos**

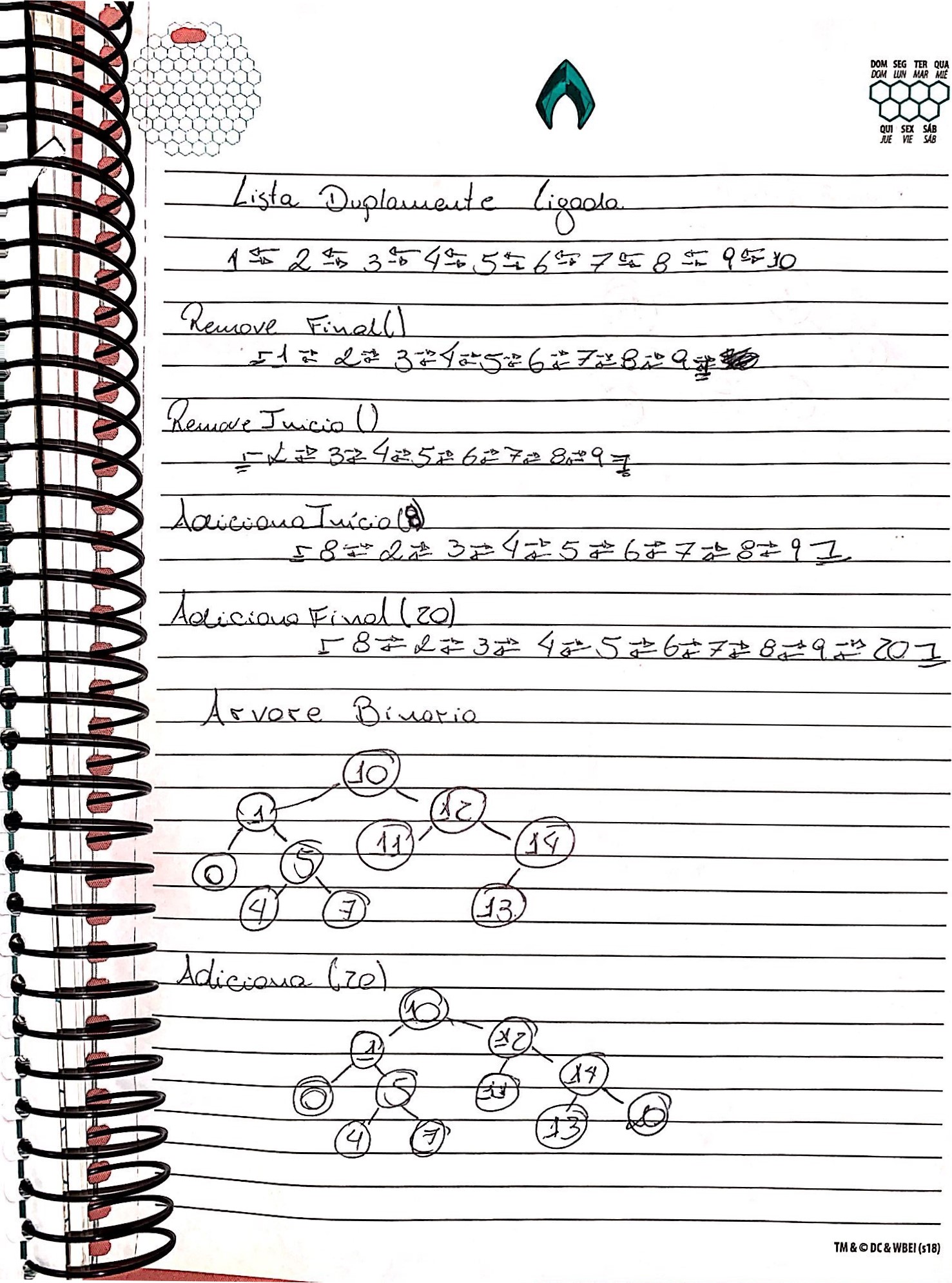
Um Grafo é uma estrutura de dados formada por um conjunto não vazio de vértices (ou nós) e por um conjunto de arestas (ou arcos), ligando estes vértices.

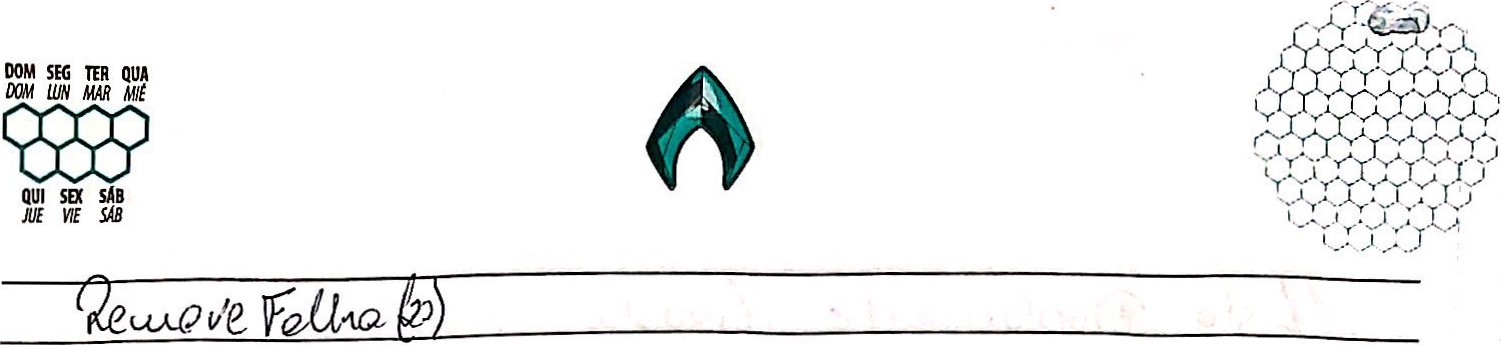
Operações:

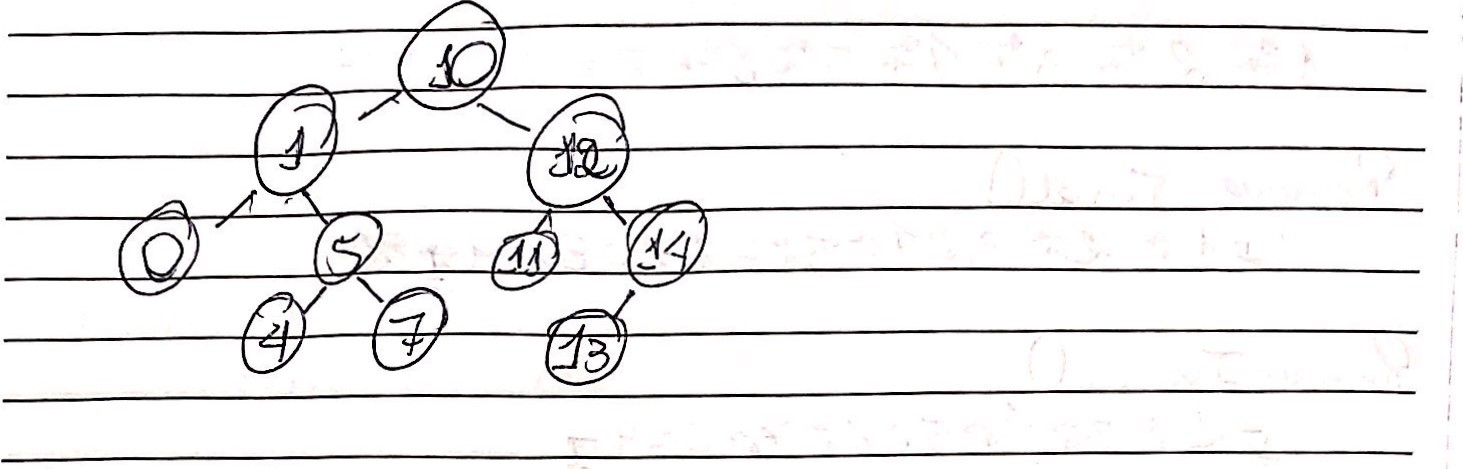
* Inserir uma aresta no grafo
* Verificar se existe determinada aresta no grafo
* Obter a lista de vértices adjacentes a determinado vértice
* Eliminar uma aresta do grafo
* Imprimir o grafo
* Obter o número de vértices do grafo
* Obter o número de arestas do grafo
* Obter a aresta de menor peso de um grafo



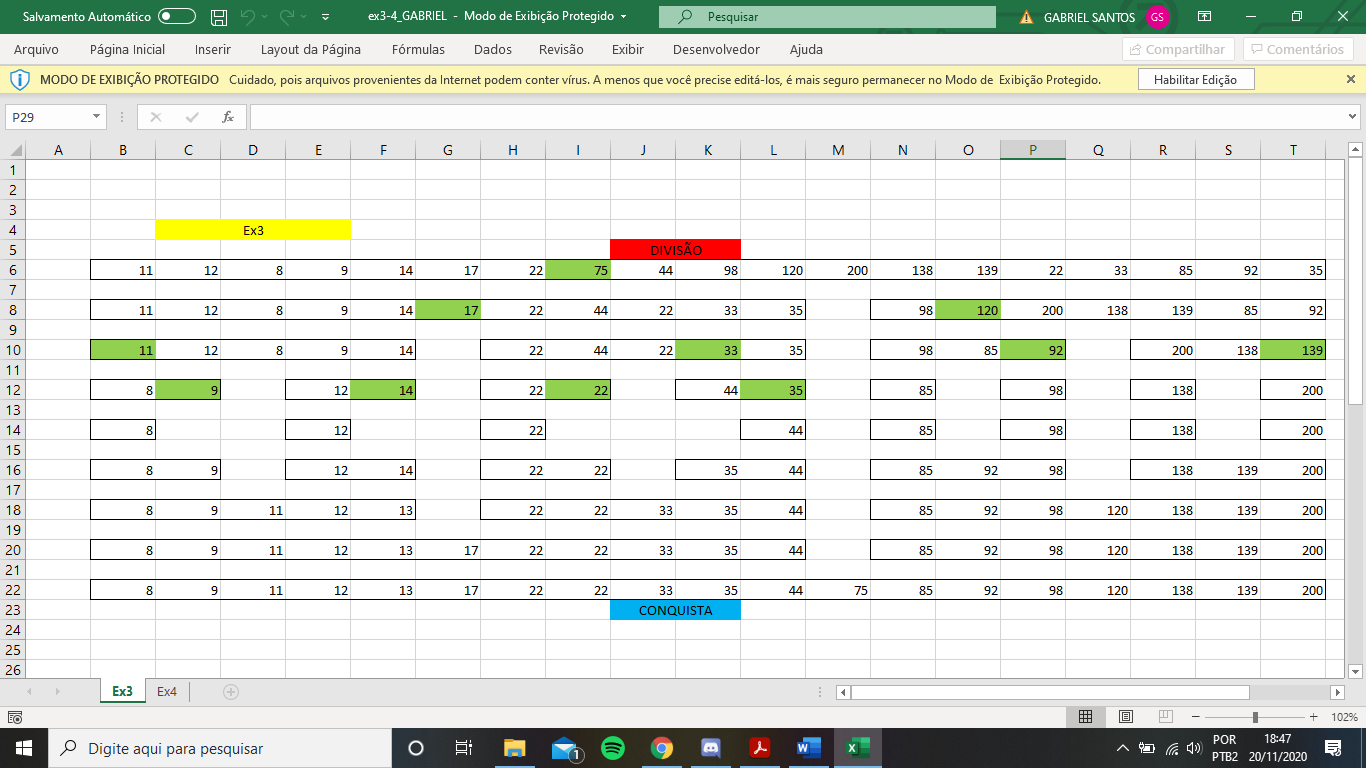




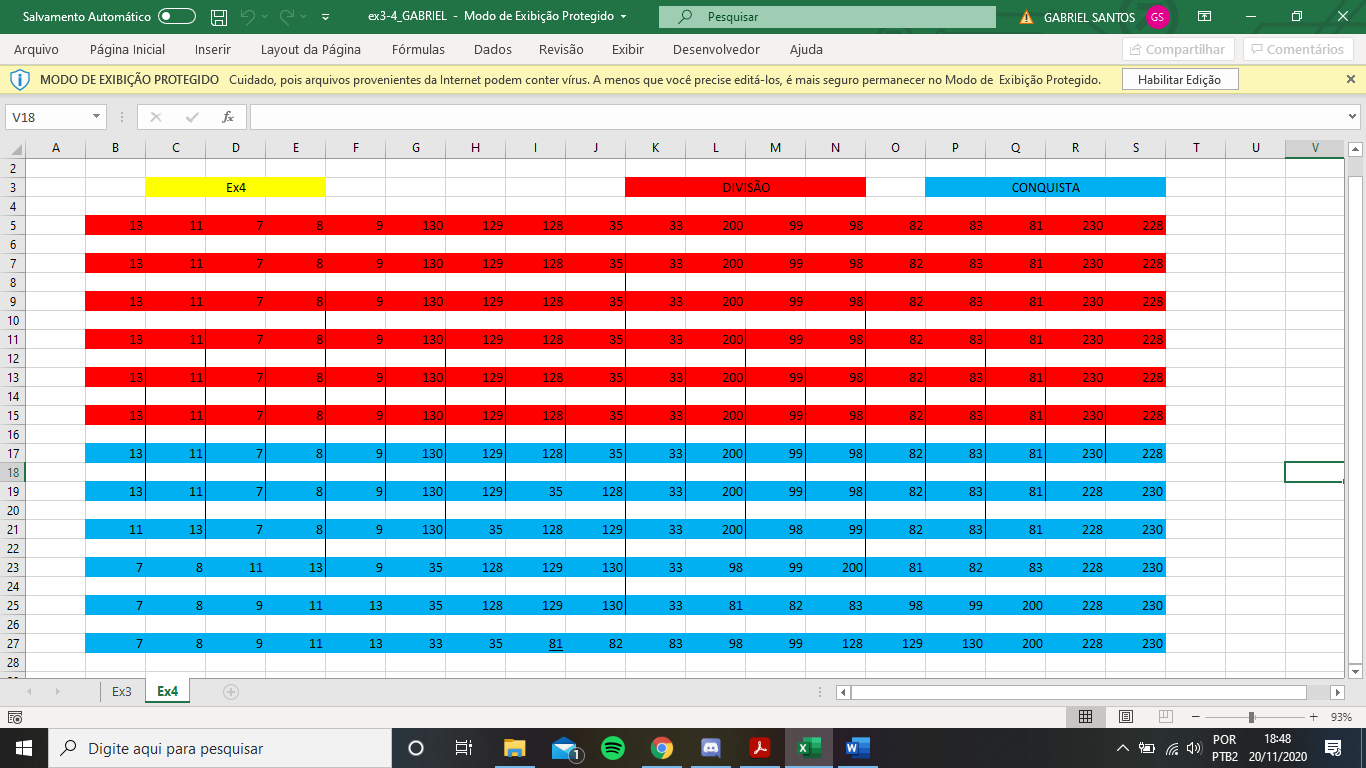




Exercício 3



Exercício 4



Exercício 5

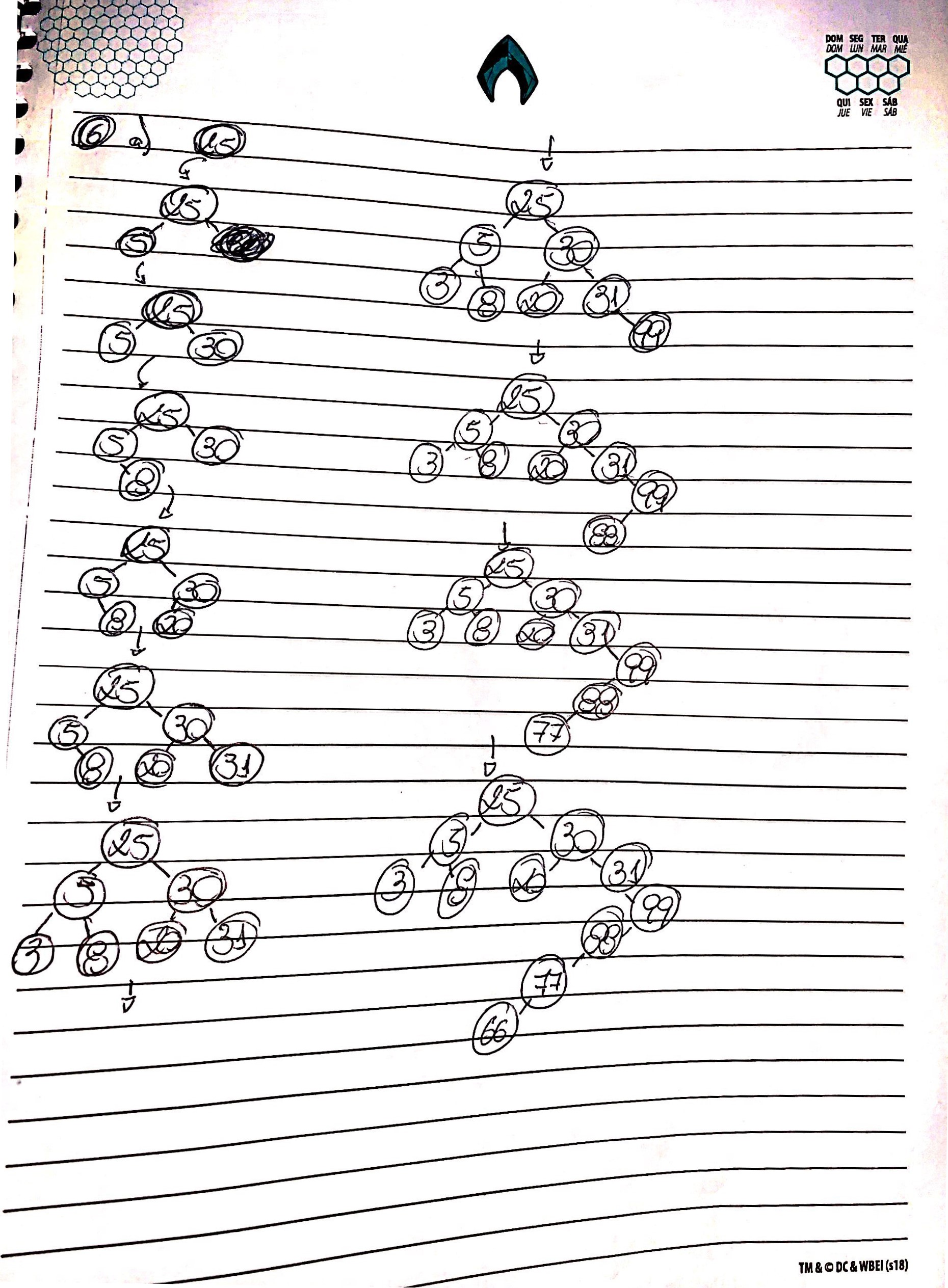
Merge Sort: Opera em duas: Primeiro, a lista é completamente separada em valores únicos. Depois da separação, entra a intercalação, onde os valores serão ordenados enquanto são juntados.

Quick Sort: separa o subconjunto em 2 de acordo com um pivô definido no programa, onde um subconjunto irá possuir valores menores que o pivô e o outro terá valores maiores. Este processo se repetirá até os subconjuntos terem apenas um valor, sobrando assim só juntá-los, pois os valores já estão ordenados.

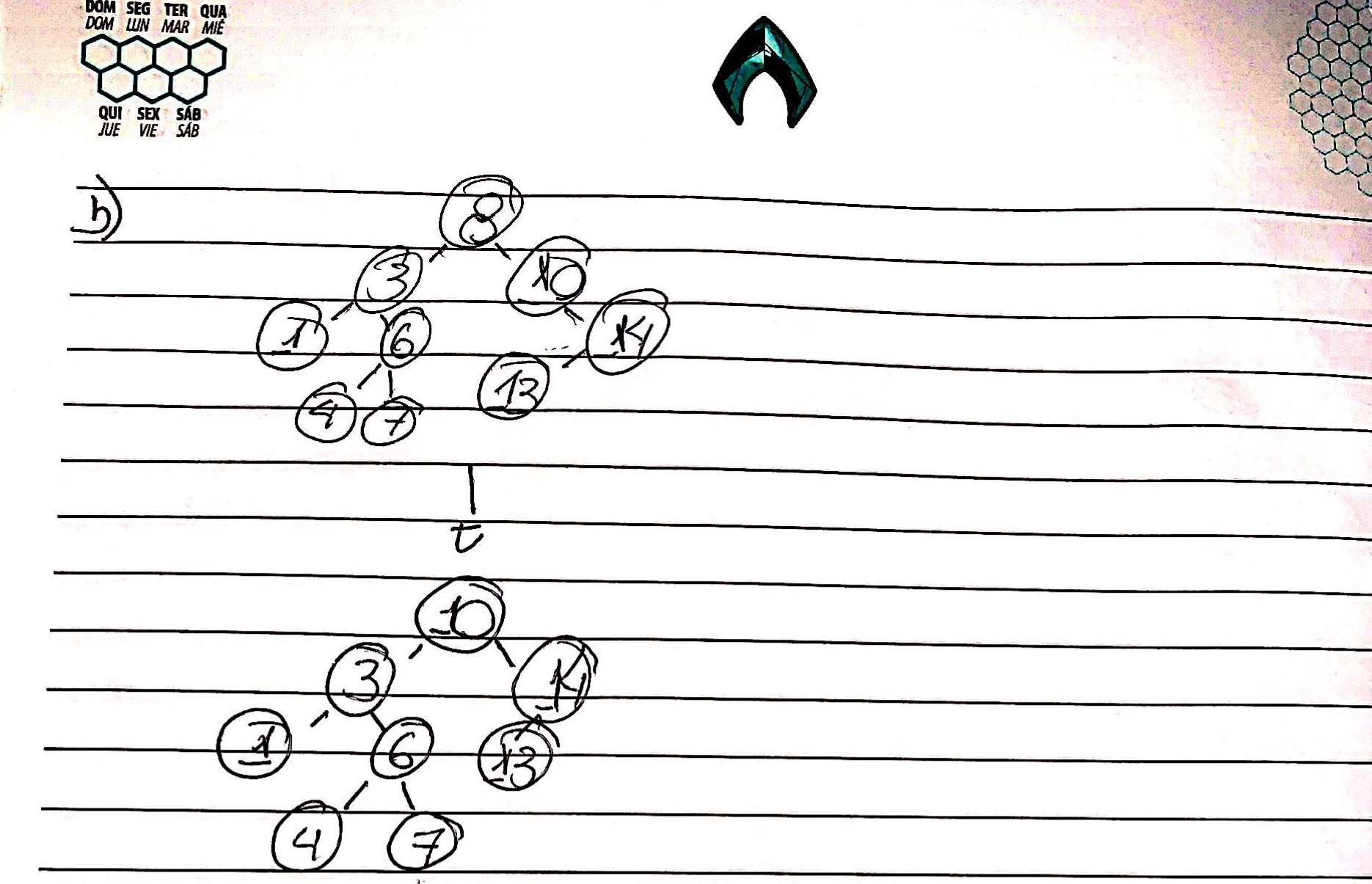
Heap sort: trabalha no conceito de árvore binária, podendo operar tanto com heap máximo (maior valor acima) quanto heap mínimo (menor valor acima). O primeiro valor da lista será considerado o raiz, sendo o menor valor da lista. Cada valor terá 2 galhos que terão de ser maiores que o mesmo. Caso um galho seja menor que a origem, há a substituição dos valores.

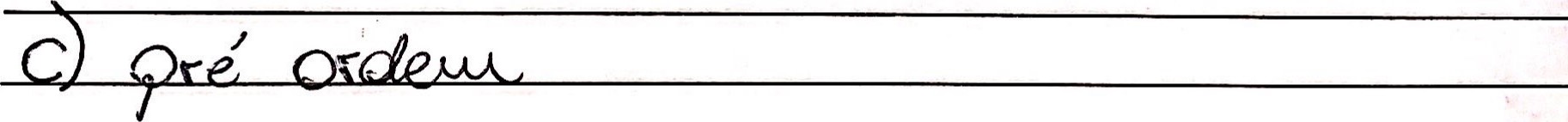
O algoritmo de ordenação apresentado é um MergeSort, em que o ordenaA é a parte da divisão do vetor em várias partes menores e o ordenaB é a parte da conquista, onde essas partes menores são organizadas e agrupadas novamente

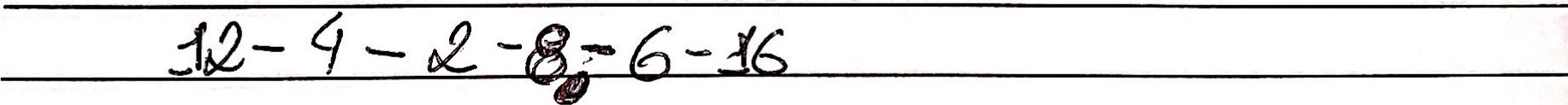
Exercício 6

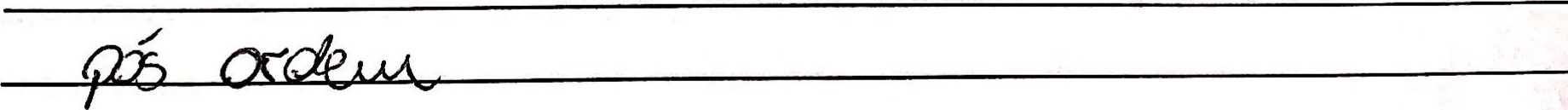


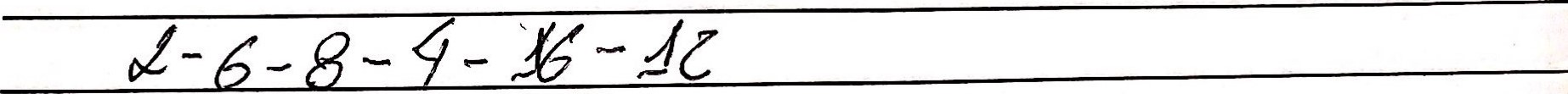
Digitalizada com CamScanner

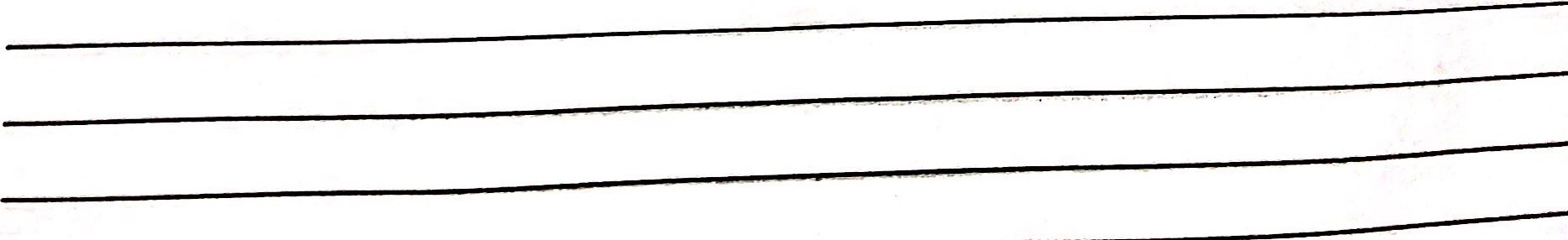












Digitalizada com CamScanner

Exercício 7

Alternativa C

Exercício 8

Alternativa C