Lista de Exercícios de Árvores e Ordenação de Dados
Heapsort

1) Considere o seguinte vetor de entrada [30 85 33 92 58 72 91], indexado por 1, 2,...,n:
    a. Utilizando o algoritmo Heapsort, rearranje os elementos do vetor para formar a representação de um *heap*, utilizando o próprio vetor de entrada. Mostre, passo a passo, os desenhos para ilustrar a formação do *heap*.
    b. A partir do *heap* criado, execute as iterações do Heapsort necessárias para ordenar o vetor. Mostre os desenhos para ilustrar, passo a passo, a ordenação.

2) Considere o seguinte vetor de entrada [19 22 31 47 79 90 35], indexado por 1, 2,...,n.
    a. Utilizando o algoritmo Heapsort, rearranje os elementos do vetor para formar a representação de um *heap*, utilizando o próprio vetor de entrada. Mostre, passo a passo, os desenhos para ilustrar a formação do *heap*.
    b. A partir do *heap* criado, execute as iterações do Heapsort necessárias para ordenar o vetor. Mostre os desenhos para ilustrar passo a passo a ordenação.

3) Considere o seguinte vetor de entrada [40 32 65 99 51 19 77], indexado por 1, 2,...,n.
    a. Utilizando o algoritmo Heapsort, rearranje os elementos do vetor para formar a representação de um *heap*, utilizando o próprio vetor de entrada. Mostre, **passo a passo**, os desenhos para ilustrar a formação do *heap*.
    b. A partir do *heap* criado, execute as iterações do Heapsort necessárias para ordenar o vetor. Mostre os desenhos para ilustrar **passo a passo** a ordenação.

4) Implemente o algoritmo de Heapsort para ordenar, em ordem decrescente, um vetor de inteiros.

5) Implemente o algoritmo de Heapsort para ordenar, em ordem crescente, um vetor de String.

6) Implemente o algoritmo de Heapsort para ordenar, em ordem decrescente de faltas, um vetor de Aluno (conforme descrição da classe Aluno abaixo)

```
public class Aluno {
    private String matr;
    private String nome;
    private double nota;
    private int faltas;

    ...
    ...
}
```