Ordenação de Palavras

Crie um programa para ordenar um conjunto de palavras pelo seu tamanho, neste caso, uma palavra é definida como uma sequência de letras, maiúsculas ou minúsculas. Palavras com apenas uma letra também deverão ser consideradas. Seu programa deve receber um conjunto de palavras e imprimir este mesmo conjunto ordenado decrescente pelo tamanho das palavras, ou seja, do maior tamanho para o menor tamanho. Se o tamanho das palavras for igual, deve-se colocar as palavras de tamanho igual em ordem lexicográfica, ou seja, como ordem do dicionário (ordem alfabética crescente).

Para ordenar a sequência de palavras o seu programa deve usar o algoritmo de ordenação **MergeSort** modificado para atender os requisitos acima.

Entrada do programa

A primeira linha da entrada possui um único inteiro **N**, que indica o número de casos de teste. Cada caso de teste estará em linha do arquivo e poderá conter várias palavras separadas por espaço em branco. Considere que em uma linha não teremos palavras repetidas, mas podemos ter palavras com somente um caractere.

Exemplos de arquivos de entrada.

```
4
Top Coder comp Wedn at midnight
one three five
I love cpp C
sj sa df r e w f d s v c x z sd fd a
```

Dicas:

- Para separar as palavras armazenadas em uma linha use método split(“ “) da classe `String`.
- Para comparar duas palavras e descobrir qual é maior que outra use o método `length()`, para verificar qual palavra vem antes da outra use o método `compareTo()` da classe `String`.
- Para converter os caracteres da uma `palavra` todos em letras minúsculas use a função `toLowerCase()` da classe `String`.
- Para saber mais sobre `Strings` acessem:
  https://www.devmedia.com.br/java-string-manipulando-metodos-da-classe-string/29862
  https://www.guj.com.br/t/metodo-compareto/334450/2

Saída do programa

A saída deve conter as palavras de uma linha da entrada ordenada pelo tamanho em uma única linha e, caso tenham o mesmo tamanho, em ordem alfabética. Um espaço em branco deve ser impresso entre as palavras e todas as palavras devem ser impressas com letras minúsculas.

Exemplo de saída

```
midnight coder comp wedn top at
three five one
love cpp c i
df fd sa sd sj a c d e f r s v w x z
```

Orientações para realização do trabalho 3

Esta trabalho pode ser feito em **dupla** ou **individualmente**, basta que somente um dos integrantes entregue um arquivo zipado (.zip) com o código fonte da solução do problema, o arquivo fonte deve conter o seguinte cabeçalho no início do arquivo.

```
/*
Entrega do Trabalho 2- Algoritmos e Programação II

Nós,

Nome completo
Nome completo

declaramos que

todas as respostas são fruto de nosso próprio trabalho,
não copiamos respostas de colegas externos à equipe,
não disponibilizamos nossas respostas para colegas externos ao grupo e
não realizamos quaisquer outras atividades desonestas para nos beneficiar ou
prejudicar outros.
*/
```

O programa deve estar bem documentado e implementado na linguagem **Java**, a entrega deve ser feita pelo **Blackboard** (não serão aceitas entregas via e-mail) e será avaliado de acordo com os seguintes critérios:

* Funcionamento do programa;
* O programa deve estar na linguagem **Java**.
* O quão fiel é o programa quanto à descrição do enunciado;
* Comentários e legibilidade do código;
* Clareza na nomenclatura de variáveis e funções.

Como esta prova pode ser em grupo (até 2 integrantes), evidentemente você pode “discutir”' o problema dado com outros grupos, inclusive as “dicas”' para chegar às soluções, mas você deve ser responsável pela solução final e pelo desenvolvimento do seu programa. Ou seja, qualquer tentativa de fraude será punida com a nota zero. Para maiores esclarecimentos leiam o documento **“Orientações para Desenvolvimento de Trabalhos Práticos”.**