Tecnologia em Análise e Desenvolvimento de Sistemas Algoritmos e Programação II		
Professor: **Alexandre Mignon**	2º Período	*Setembro/2021*

Orientações para realização do trabalho 1

Este trabalho pode ser feito em grupo de no **máximo três integrantes**, 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
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 este trabalho pode ser em grupo (até 3 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”.**

Primeiro Dicionário do Samuel

Samuel de apenas 8 anos tem um sonho - ele deseja criar o seu próprio dicionário. Isto não é uma tarefa fácil para ele, pois conhece poucas palavras. Bem, ao invés de pensar nas palavras que sabe, ele teve uma ideia brilhante. A partir do seu livro de histórias favorito, ele vai criar um dicionário com todas as palavras distintas que existem nele. Ordenando estas palavras em ordem alfabética, o trabalho estará feito. É claro, isso é uma tarefa que toma um certo tempo e, portanto, a ajuda de um programador de computador como você é muito bem-vinda.

Você foi convidado a escrever um programa que liste todas as diferentes palavras que existem em um arquivo texto. Neste caso, uma palavra é definida como uma sequência de letras, maiúsculas ou minúsculas. Palavras com apenas uma letra também deverão ser consideradas. Além disso o seu programa deverá ser "CaSe InSeNsItIvE". Por exemplo, palavras como "Apple", "apple" ou "APPLE" deverão ser consideradas como mesma palavra.

Para garantir que não tenhamos palavras repetidas no dicionário do Samuel, para cada palavra lida no arquivo texto deve feita a busca no dicionário (usando a **busca binária**), caso a palavra já conste no dicionário a palavra lida deve ser descartada, caso contrário a palavra deverá ser inserida no dicionário (vetor) de **forma ordenada**, essa operação deve gastar no **máximo N passos** para cada palavra nova. **Importante**, não é para inserir todas as palavras no vetor de palavras e depois ordenar, e sim a cada palavra nova, esta deve ser inserida no vetor de palavras que continuará ordenado.

Entrada do programa

O arquivo de entrada contém várias linhas de texto, cada uma delas com várias palavras, considere que na entrada teremos no máximo 1000 palavra diferentes.

Exemplos de arquivos de entrada.

```
texT

Adventures in Disneyland

Two blondes were going to Disneyland when they came to a fork in the road The sign read disneyland LEFT

So they went home
```

Dicas:

- O dicionário será um **vetor** de `Strings` com até 1000 posições. Não é permito o uso de classes prontas de lista do Java tais como ArrayList.
- Para separar as palavras armazenadas em uma linha use método `split(“ “)` da classe `String`, cuidado que o `split` pode gerar palavras vazias “”.
- Para comparar duas `Strings` e descobrir qual vem antes da outra use o método `compareTo()` da classe `String`
- Para converter os caracteres da `String` todos em letras minúsculas use a função `toLowerCase()`
- 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

Você deve imprimir uma lista de diferentes palavras que aparecem no texto, uma palavra por linha. Todas as palavras devem ser impressas com letras minúsculas, em ordem alfabética. No final da lista você deve informar quantas palavras diferentes existe no texto

Exemplo de saída

```
a
adventures
blondes
came
disneyland
fork
going
home
in
left
read
road
sign
so
text
the
they
to
two
went
were
when
total de palavras diferentes no dicionario=22
```