PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e Informática
Trabalho Prático 2

Cursos	: *Engenharia de Computação*
	Sistemas de Informação
Disciplina	: *Algoritmos e Estruturas de Dados*
Professora	: *Eveline Alonso Veloso*

Regras Básicas:

1. Estude bastante cada par de entrada/saída.
2. Todos os programas deverão ser desenvolvidos na linguagem de programação Java.
3. Esse trabalho prático poderá ser desenvolvido em grupos de, no máximo, três integrantes.
4. Cópias de trabalho, se existirem, serão encaminhadas ao colegiado de coordenação didática do curso.
5. Fique atento ao *charset* dos arquivos de entrada e saída. Recomenda-se a utilização dos métodos da classe MyIO.java para leitura de dados do teclado.
6. Para cada exercício, vocês devem submeter apenas um arquivo (.java) por grupo. Essa regra será necessária para a submissão de trabalhos no VERDE e no identificador de plágios utilizado na disciplina.
7. A resolução (código) de cada exercício deverá ser submetida ao VERDE.
8. A correção será realizada automaticamente pelo VERDE e validada por meio de apresentações durante as aulas práticas da disciplina.

Base de Dados:

Spotify é um serviço de *streaming* de música, *podcast* e vídeo lançado oficialmente em 7 de outubro de 2008. É atualmente o serviço de *streaming* mais popular e usado do mundo. Desenvolvido pela *startup* Spotify AB (Estocolmo, Suécia), fornece conteúdo protegido de restrições relacionadas a direitos digitais de gravadoras e empresas de mídia. O Spotify é um serviço *freemium*; com recursos básicos gratuitos, com propagandas ou limitações, enquanto recursos adicionais, como qualidade de transmissão aprimorada e *downloads* de música, oferecidos para assinaturas pagas.

Spotify está disponível na maior parte da Europa, parte da América, Austrália, Nova Zelândia e parte da Ásia. Disponível para a maioria dos dispositivos modernos, incluindo computadores Windows, macOS e Linux, bem como *smartphones* e *tablets* com iOS, Windows Phone e Android. Pode-se encontrar as músicas desejadas por navegação ou pesquisas referentes a artista, álbum, gênero, lista de reprodução ou gravadora. Seus usuários podem criar, editar ou compartilhar *playlists*, compartilhar

faixas em redes sociais ou fazer *playlists* com outros usuários. Fornece acesso a mais de 30 milhões de músicas e, em julho de 2019, contava com mais de 232 milhões de usuários ativos, incluindo 108 milhões de assinantes pagantes.

O arquivo dataAEDs.csv contém um conjunto de dados relacionados a mais de 175.000 músicas coletadas da plataforma Spotify Web API, que podem ser agrupadas por artista, ano ou gênero. Tal arquivo deve ser copiado para a pasta /tmp.

Exercícios:

1. **Impressão aleatória de dados de músicas**
   Crie uma classe *Musica* com os atributos privados: *id* (String), *name* (String), *key* (String), *artists* (String[]), *release_date* (Date), *acousticness* (double), *danceability* (double), *energy* (double), *duration_ms* (int), *instrumentalness* (double), *valence* (double), *popularity* (int), *time* (double), *liveness* (double), *loudness* (double), *speechiness* (double), *year* (int).
   Sua classe também terá, pelo menos, dois construtores, e os métodos *gets*, *sets*, clone, ler e imprimir. O método imprimir exibe o valor de atributos do objeto (observe o formato de cada linha da saída esperada) e o ler lê os atributos de um objeto.
   Neste exercício, você deve preencher um vetor de objetos da classe *Musica* com os dados das diversas músicas presentes no arquivo dataAEDs.csv. Atenção para esse arquivo de entrada, pois em alguns registros o campo *release_date* está incompleto. Nesses casos, esses registros devem ser preenchidos com o *valor* "01", tanto no caso da ausência de valor para o dia quanto para o mês.
   Em seguida, seu programa deve ler a entrada padrão, que é composta por várias linhas e cada uma contém uma *string* indicando o *id* da música cujos dados devem ser exibidos na saída padrão. A última linha da entrada contém a palavra FIM.
   Na saída padrão, para cada linha de entrada, escreva uma linha com os dados do registro correspondente.

   A saída padrão deve obedecer o seguinte formato:

   id ## [artists] ## name ## realease date ## acousticness ## danceability ## instrumentalness ## liveness ## loudness ## speechiness ## energy ## duration_ms

2. **Ordenação por seleção**
   Utilizando vetores, ordene os registros das músicas do Spotify aplicando o algoritmo de ordenação por seleção, considerando que a chave de pesquisa seja o atributo ***name***. Em caso de empate, o segundo critério de ordenação deve ser o atributo ***id*** da música.
   A entrada padrão é composta por várias linhas e cada uma contém uma *string* indicando o *id* da música cujos dados devem ser inseridos no vetor de músicas a ser ordenado. A última linha da entrada contém a palavra FIM.

A saída padrão corresponde aos registros ordenados, um por linha. Em cada linha da saída, escreva os dados do registro correspondente.
Além disso, crie um arquivo de *log* na pasta corrente com o nome matrícula_selecao.txt com uma única linha contendo: seu número de matrícula, tempo de execução de seu algoritmo de ordenação (em milissegundos), número de comparações realizadas entre os elementos do vetor de músicas e número de movimentações realizadas entre os elementos do vetor. Todas as informações desse arquivo de *log* devem ser separadas por uma tabulação '\t'.

3. **Ordenação por inserção**
   Repita a questão de **Ordenação por seleção em Java**, contudo, aplicando o algoritmo de ordenação por inserção, considerando como chave de pesquisa o atributo ***id***.
   O nome do arquivo de *log* dessa questão será matrícula_insercao.txt.

4. ***Heapsort***
   Repita a questão de **Ordenação por seleção em Java**, contudo, aplicando o algoritmo de ordenação *heapsort*, considerando como chave de pesquisa o atributo ***release_date***. Em caso de empate, o segundo critério de ordenação deve ser o atributo ***name*** da música.
   O nome do arquivo de *log* dessa questão será matrícula_heapsort.txt.

5. ***Mergesort***
   Repita a questão de **Ordenação por seleção em Java**, contudo, aplicando o algoritmo de ordenação *mergesort*, considerando como chave de pesquisa o atributo ***energy***. Em caso de empate, o segundo critério de ordenação deve ser o atributo ***name*** da música.
   O nome do arquivo de *log* dessa questão será matrícula_mergesort.txt.

6. ***Quicksort***
   Repita a questão de **Ordenação por seleção em Java**, contudo, aplicando o algoritmo de ordenação *quicksort*, considerando como chave de pesquisa o atributo ***duration***. Em caso de empate, o segundo critério de ordenação deve ser o atributo ***name*** da música.
   O nome do arquivo de *log* dessa questão será matrícula_quicksort.txt.

7. **Bolha**
   Repita a questão de **Ordenação por seleção em Java**, contudo, aplicando o algoritmo de ordenação *bubblesort*, considerando como chave de pesquisa o atributo ***danceability***. Em caso de empate, o segundo critério de ordenação deve ser o atributo ***name*** da música.
   O nome do arquivo de *log* dessa questão será matrícula_bolha.txt.