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

Curso	: *Engenharia de Software*
Disciplina	: *Algoritmos e Estruturas de Dados II*
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 especificada em seu enunciado (C ou 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 ou .cpp). 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:

A *National Basketball Association* (em português: Associação Nacional de Basquetebol; abreviação oficial: NBA) é a principal liga de basquetebol profissional da América do Norte. Com 30 franquias como membros (29 nos Estados Unidos e 1 no Canadá), a NBA também é considerada a principal liga de basquete do mundo. É um membro ativo da USA *Basketball* (USAB), que é reconhecida pela FIBA (a Federação Internacional de Basquetebol) como a entidade máxima e organizadora do basquetebol nos Estados Unidos. A NBA é uma das 4 '*major leagues*' de esporte profissional na América do Norte. Os jogadores da NBA são os esportistas mais bem pagos do mundo, por salário médio anual.

A liga foi fundada na cidade de Nova Iorque, em 6 de junho de 1946, como a *Basketball Association of America* (BAA). Adotou o nome de *National Basketball Association* em 1949, quando se fundiu com a rival *National Basketball League* (NBL). A liga tem diversos escritórios ao redor do mundo, além de vários dos

próprios clubes fora da sede principal na *Olympic Tower*, localizada na Quinta Avenida, 645. Os estúdios da NBA *Entertainment* e da NBA TV estão localizados em Secaucus, New Jersey.

O arquivo players.csv contém um conjunto de dados de jogadores da liga de basquete norte-americana – NBA – extraídos do *site* https://www.kaggle.com/drgilermo/nba-players-stats. Essa base contém registros de jogadores desde 1950, um total de 67 temporadas da NBA. Esse arquivo sofreu algumas adaptações para ser utilizado neste e nos próximos trabalhos práticos da disciplina. Tal arquivo deve ser copiado para a pasta /tmp/.

Exercícios:

1. **Impressão aleatória de dados de jogadores em Java**
   Crie uma classe *Jogador* com os atributos privados: *id* (int), *nome* (String), *altura* (int), *peso* (int), *universidade* (String), *anoNascimento* (int), *cidadeNascimento* (String), e *estadoNascimento* (String). Sua classe também terá, pelo menos, dois construtores, e os métodos *gets*, *sets*, clone e imprimir. O método imprimir exibe o valor de todos os atributos do objeto (observe o formato de cada linha da saída esperada).
   Neste exercício, você deve preencher um vetor de objetos da classe *Jogador* com os dados dos diversos jogadores presentes no arquivo players.csv. Atenção para esse arquivo de entrada, pois em alguns registros faltam valores e esses devem ser substituídos pela *string* "nao informado", na saída padrão.
   Em seguida, seu programa deve ler a entrada padrão, que é composta por várias linhas e cada uma contém uma *string* indicando o *id* do jogador 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 todos os dados do registro correspondente.

2. **Pesquisa sequencial em Java**
   Faça a inserção dos registros correspondentes a alguns jogadores em um vetor e, em seguida, faça algumas pesquisas sequenciais. A chave primária de pesquisa será o atributo ***nome***.
   A entrada padrão é dividida em duas partes. A primeira contém, em cada linha, uma *string* indicando o *id* do jogador cujos dados devem ser inseridos no vetor de jogadores. Após a palavra FIM, inicia-se a segunda parte da entrada padrão, que também é composta por várias linhas, sendo que cada uma possui o nome de um jogador que deve ser pesquisado no vetor de jogadores. A última linha dessa parte também terá a palavra FIM.
   A saída padrão será composta por várias linhas contendo as palavras SIM/NAO para indicar se cada um dos nomes pesquisados existe ou não no vetor de jogadores.
   Além disso, crie um arquivo de *log* na pasta corrente com o nome matrícula_sequencial.txt com uma única linha contendo: seu número de matrícula, tempo de execução de seu algoritmo (em milissegundos) e número de

comparações realizadas. Todas as informações desse arquivo de *log* devem ser separadas por uma tabulação '\t'.

3. **Ordenação por seleção em Java**
   Utilizando vetores, ordene os registros dos jogadores da NBA aplicando o algoritmo de ordenação por seleção, considerando que a chave de pesquisa seja o atributo ***nome***.
   A entrada padrão é composta por várias linhas e cada uma contém uma *string* indicando o *id* do jogador cujos dados devem ser inseridos no vetor de jogadores 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 de saída, escreva todos 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 jogadores e número de movimentações realizadas entre os elementos do vetor. Todas as informações desse arquivo de *log* devem ser separadas por uma tabulação '\t'.

4. **Ordenação por inserção em Java**
   Repita a questão de **Ordenação por seleção em Java**, contudo, aplicando o algoritmo de ordenação por inserção, considerando como chave de pesquisa o atributo ***anoNascimento***. Em caso de empate, o segundo critério de ordenação deve ser o atributo *nome* do jogador.
   O nome do arquivo de *log* dessa questão será matrícula_insercao.txt.

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

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

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

8. ***Shellsort* em Java**

Repita a questão de **Ordenação por seleção em Java**, contudo, aplicando o algoritmo de ordenação *shellsort*, considerando como chave de pesquisa o atributo ***peso***. Em caso de empate, o segundo critério de ordenação deve ser o atributo *nome* do jogador.
O nome do arquivo de *log* dessa questão será matrícula_shellsort.txt.

9. Bolha em Java

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

10. Impressão aleatória de dados de jogadores em C

Crie uma *struct Jogador* com os seguintes membros: *id* (int), *nome* (*string*), *altura* (int), *peso* (int), *universidade* (*string*), *anoNascimento* (int), *cidadeNascimento* (*string*), e *estadoNascimento* (*string*). Implemente também, pelo menos, sub-rotinas (funções ou procedimentos) para: criar um novo jogador, clonar um jogador informado e imprimir os dados de um jogador específico. A sub-rotina *imprimir* exibe o valor de todos os membros do jogador (observe o formato de cada linha da saída esperada).
Neste exercício, você deve preencher um vetor de jogadores com os dados dos diversos jogadores presentes no arquivo players.csv. Atenção para esse arquivo de entrada, pois em alguns registros faltam valores e esses devem ser substituídos pela *string* "nao informado", na saída padrão.
Em seguida, seu programa deve ler a entrada padrão, que é composta por várias linhas e cada uma contém uma *string* indicando o *id* do jogador 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 todos os dados do registro correspondente.

11. Pesquisa binária em C

Repita a questão **Pesquisa sequencial em Java**, contudo, aplicando pesquisa binária implementada recursivamente, na linguagem C.
O nome do arquivo de *log* desta questão será matrícula_binaria_C.txt.
A entrada desta questão não está ordenada.

12. Bolha em C

Repita a questão **Bolha em Java**, contudo, implementando o algoritmo de ordenação *bubblesort* na linguagem C, considerando como chave de pesquisa o atributo ***anoNascimento***. Em caso de empate, o segundo critério de ordenação deve ser o atributo *nome* do jogador.
O nome do arquivo de *log* dessa questão será matrícula_bolha_C.txt.

13. Seleção recursiva em C

Repita a questão de **Ordenação por seleção em Java**, contudo, implementando o algoritmo de ordenação por seleção na linguagem C, de forma

recursiva, considerando como chave de pesquisa o atributo ***cidadeNascimento***. Em caso de empate, o segundo critério de ordenação deve ser o atributo *nome* do jogador.
O nome do arquivo de *log* dessa questão será matrícula_selecaoRecursiva_C.txt.