PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Instituto de Ciências Exatas e Informática
Trabalho Prático 3

Curso : *Engenharia de Software*
Disciplina : *Algoritmos e Estruturas de Dados II*
Professora : *Eveline Alonso Veloso*

Regras Básicas:

1. Desenvolva esse trabalho prático a partir do que já foi implementado no Trabalho prático 2.
2. Estude bastante cada par de entrada/saída.
3. Todos os programas deverão ser desenvolvidos na linguagem de programação especificada em seu enunciado (C ou Java).
4. Esse trabalho prático poderá ser desenvolvido em grupos de, no máximo, três integrantes.
5. Cópias de trabalho, se existirem, serão encaminhadas ao colegiado de coordenação didática do curso.
6. 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.
7. Para cada exercício, vocês devem submeter apenas um arquivo (.java ou .cpp) por grupo. Essa regra será necessária para a submissão de trabalhos no VERDE e no identificador de plágios utilizado na disciplina.
8. A resolução (código) de cada exercício deverá ser submetida ao VERDE.
9. 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:

Estruturas de dados implementadas por meio de vetores:

1. **Pilha implementada por meio de vetor em Java**
   Crie uma pilha, implementada por meio de vetor, de objetos da classe *Jogador*. Lembre-se que, na verdade, temos um vetor de referências para objetos do tipo *Jogador*.
   Neste exercício, faremos inserções e remoções de itens na pilha e, após o processamento de todas as operações, mostraremos seus elementos.

   Os métodos de sua pilha devem operar conforme descrito a seguir, respeitando-se parâmetros e tipos de retorno:
   - Sua classe *Pilha* deverá ter dois construtores.
   - *void empilhar*(*Jogador jogador*): empilha um objeto do tipo *Jogador*.
   - *Jogador desempilhar* (): desempilha e retorna o *Jogador* do topo da pilha.
   - *void mostrar*(): a partir do fundo da pilha, para todos os objetos do tipo *Jogador* presentes na pilha, exibe a posição do objeto na pilha seguida dos valores de todos os seus atributos (observe o formato de cada linha da saída esperada).

   A entrada padrão é dividida em duas partes. A primeira contém, em cada linha, uma *string* indicando o *id* do jogador que deve ser inicialmente inserido na pilha de jogadores, na ordem em que são apresentados. Após a palavra FIM, inicia-se a segunda parte da entrada padrão.
   A primeira linha dessa segunda parte da entrada padrão apresenta um número inteiro *n* indicando a quantidade de jogadores que serão em seguida empilhados ou desempilhados. Nas próximas *n* linhas, tem-se *n* comandos de inserção ou remoção que devem ser processados neste exercício. Cada uma dessas linhas tem uma palavra de comando, conforme descrito a seguir:
   - I: empilhar;
   - R: desempilhar.

   No caso dos comandos de inserção, temos também uma *string* indicando o *id* do jogador que deve empilhado na pilha de jogadores.

A saída padrão apresenta uma linha para cada jogador desempilhado, sendo que essa informação será constituída pela *string* "(R)" seguida do atributo *nome* do jogador retirado da pilha. Em seguida, teremos, ainda na saída padrão, todos os atributos relativos aos jogadores armazenados na pilha após o processamento de todas as operações de inserção e remoção (observe o formato de cada linha da saída esperada).

2. **Fila circular implementada por meio de vetor em Java**

Crie uma fila circular, implementada por meio de vetor, de objetos da classe *Jogador*. Essa fila deve conseguir armazenar simultaneamente até cinco jogadores. Neste exercício, faremos inserções e remoções de itens na fila.

Os métodos de sua fila circular devem operar conforme descrito a seguir, respeitando-se parâmetros e tipos de retorno:

- Sua classe *Fila* deverá ter dois construtores.
- *void enfileirar* (*Jogador jogador*): enfileira um objeto do tipo *Jogador*.
- *Jogador desenfileirar* (): desenfileira e retorna o *Jogador* da frente da fila.
- *void mostrar*(): para todos os objetos do tipo *Jogador* presentes na fila, exibe a posição do objeto na fila seguida dos valores de todos os seus atributos (observe o formato de cada linha da saída esperada).
- *double obterMediaAltura* (): calcula e retorna a média das alturas dos itens presentes na fila.

A entrada padrão será como a da questão **Pilha implementada por meio de vetor em Java**, contudo, o comando I será utilizado para inserir na fila (enfileirar); e R, para remover da fila (desenfileirar).
Observe que, quando, no momento de execução da operação enfileirar, a fila estiver cheia, antes de enfileirar um jogador será necessário desenfileirar outro.

A saída padrão será um número **inteiro** corresponde à média **arredondada** das alturas dos itens contidos na fila, após cada inserção.
Além disso, a saída padrão também apresenta uma linha para cada jogador desenfileirado, sendo que essa informação será constituída pela *string* "(R)" seguida do atributo *nome* desse jogador. Em seguida, teremos, ainda na saída padrão, todos os atributos relativos aos jogadores armazenados na fila após o processamento de todas as operações de inserção e remoção (observe o formato de cada linha da saída esperada).

3. **Lista implementada por meio de vetor em Java**

Crie uma lista, implementada por meio de vetor, de objetos da classe *Jogador*.
Neste exercício, faremos inserções e remoções de itens na lista e, após o processamento de todas as operações, mostraremos seus elementos.

Os métodos de sua lista devem operar conforme descrito a seguir, respeitando-se parâmetros e tipos de retorno:

- Sua classe *Lista* deverá ter dois construtores.
- *void inserirInicio*(*Jogador jogador*): insere um objeto do tipo *Jogador* na primeira posição da lista, necessitando remanejar todos os demais.

- *void inserir*(*Jogador jogador*, *int posicao*): insere um jogador na posição da lista indicada pelo parâmetro *posicao*, desse método; onde 0 <= *posicao* <= *n*, sendo *n* o número de jogadores já inseridos na estrutura. Esse método também remaneja os demais objetos da lista.
- *void inserirFim*(*Jogador jogador*): insere um objeto da classe *Jogador* na última posição da lista.
- *Jogador removerInicio*(): remove e retorna o primeiro jogador da lista, remanejando os demais.
- *Jogador remover*(*int posicao*): remove e retorna o objeto *Jogador* armazenado na posição da lista indicada pelo parâmetro *posicao*, desse método; necessitando remanejar os demais.
- *Jogador removerFim*(): remove e retorna o último *Jogador* da lista.
- *void mostrar*(): para todos os objetos do tipo *Jogador* presentes na lista, exibe a posição do objeto na lista seguida dos valores de todos os seus atributos (observe o formato de cada linha da saída esperada).

A entrada padrão é dividida em duas partes, conforme a entrada padrão do exercício **Pilha implementada por meio de vetor em Java**. A primeira contém, em cada linha, uma *string* indicando o *id* do jogador que deve ser inicialmente inserido na lista de jogadores, na ordem em que são apresentados. Após a palavra FIM, inicia-se a segunda parte da entrada padrão.
A primeira linha dessa segunda parte da entrada padrão apresenta um número inteiro *n* indicando a quantidade de jogadores que serão em seguida inseridos ou removidos da lista. Nas próximas *n* linhas, tem-se *n* comandos de inserção ou remoção que devem ser processados neste exercício. Cada uma dessas linhas tem uma palavra de comando, conforme descrito a seguir:

- II: inserir no início;
- I* inserir em uma determinada posição;
- IF: inserir no final;
- RI: remover do início;
- R*: remover de uma determinada posição; e
- RF: remover no final.

No caso dos comandos de inserção, temos também uma *string* indicando o *id* do jogador que deve inserido na lista de jogadores.
No caso dos comandos de inserção e remoção "em uma determinada posição", temos também um inteiro indicando essa posição. No comando de inserção, a posição fica imediatamente após a palavra de comando. Lembre-se que o primeiro item da lista encontra-se na posição 0.

A saída padrão deve ser como a da questão **Pilha implementada por meio de vetor em Java**.

4. Pilha implementada por meio de vetor em C

Refaça o exercício **Pilha implementada por meio de vetor em Java** na linguagem de programação C.

Estruturas de dados implementadas por meio de células auto-referenciadas:

5. **Pilha com alocação dinâmica de memória em Java**
   Refaça o exercício **Pilha implementada por meio de vetor em Java** usando alocação dinâmica de memória.
   Neste exercício, sua classe *Pilha* deverá ter apenas um construtor.

6. **Fila com alocação dinâmica de memória em Java**
   Refaça o exercício **Fila circular implementada por meio de vetor em Java** usando alocação dinâmica de memória.
   Neste exercício, sua classe *Fila* deverá ter apenas um construtor.
   Lembre-se que essa fila deve conseguir armazenar simultaneamente, no máximo, cinco jogadores.

7. **Lista encadeada em Java**
   Refaça o exercício **Lista implementada por meio de vetor em Java** usando lista encadeada.
   Neste exercício, sua classe *Lista* deverá ter apenas um construtor.

8. ***Quicksort* com lista duplamente encadeada em Java**
   Refaça o exercício ***Quicksort* em Java** do **Trabalho prático 2** usando lista duplamente encadeada.
   O nome do arquivo de *log* dessa questão será matrícula_quicksort2.txt.

9. **Fila com alocação dinâmica de memória em C**
   Refaça o exercício **Fila com alocação dinâmica de memória em Java** na linguagem de programação C.