METROCAMP Grupo ibmec	**Avaliação Prática 2 (v 1.0)** Curso de Eng. da Computação Metrocamp
Nome:	Turma:
Professor: Giulliano Paes Carnielli	Data:

Avaliação Prática de Estrutura de Dados: Projeto Jogo de Xadrez

Instruções:

1. Esta atividade deve ser desenvolvida em grupos de 5 ou 6 alunos.
2. Todo grupo deve ter um nome único.
3. A entrega deve ser feita, por email, como um pacote ZIP contendo um projeto do Code::Blocks.
4. O título do email deve ter o formato: `[ED] - Projeto Pratico <2/2> - <nome do grupo>`
5. Um trabalho estará sujeito a anulação imediata caso: não compile, não apresente minimamente as funções esperadas.
6. Os nomes de todos os membros do grupo devem aparecer no cabeçalho de arquivos de código e no corpo do email usado para entregar a avaliação.
7. O código deve ser organizado e bem documentado, com o risco de decréscimo na nota.
8. O plágico ocasionará a anulação de todas as avaliações envolvidas.
9. **BOM TRABALHO.**

1. Jogo de Xadrez:

Incluiremos novas fucionalidades ao Jogo de Xadrez simplificado, desenvolvido na primeira etapa deste projeto.

Os grupos devem observar as especificações fornecidas neste documento. Qualquer variação deve ser discutida com o professor, aprovada e documentada.

2. Pré-Requisitos:

Esta etapa baseia-se no projeto construído no primeiro bimestre. Portanto, para que as novas funcionalidades possam ser adicionadas conforme as especificações deste documento, algums pré-requisitos devem ser atendidos:

a. Cada grupo deve ter uma versão funcional do programa "Jogo de Xadrez"
b. Observações feitas pelo professor, durante a avaliação da 1a Etapa, devem ter sido implementadas
c. O programa deve ser estruturado conforme as especificações da 1a parte do projeto
d. O TAD Jogo deve possuir um construtor (criaJogo) que cria todas as peças do jogo, armazenando-as em uma Lista de Peças. Depois cria o Tabuleiro.
e. O TAD Peca deve armazenar as coordenadas atuais de uma peça no tabuleiro
f. O TAD Tabuleiro deve possuir um construtor (criaTabuleiro) que estabelece a configuração inicial do tabuleiro de acordo com uma lista de peças passada como parâmetro

3. Preparação:

Para viabilizar a implementação das novas funcionalidade, as seguintes modificações são necessárias.

TAD Jogada (Nova):

Este TAD deve ser usado para registrar cada jogada em uma partida. Deve ter os seguintes atributos:

- Peca *pecaMovida: ponteiro para a peça que foi movida na jogada
- Peca *pecaCapturada: ponteiro para uma peça capturada na jogada (se não houver captura, este ponteiro deve apontar para NULL)
- Coordenada Origem: coordenada de origem da jogada (x, y)
- Coordenada Destino: coordenada de destino da jogada (x, y)
- double tempoJogada: armazena o tempo decorrido até que o jogador tenha realizado sua jogada

TAD PilhaJogada (Nova):

Este TAD deve ser construído como uma pilha para objetos Jogada (acima). Sua implementação deve ser semelhante à fornecida em sala, com a difereça de que terá um ponteiro para Jogada como dado (ao invés de um int ou float).

TAD Jogo: um novo atributo deve ser acrescentado para armazenar jogadas.

- PilhaJogada * jogadas: ponteiro para o *header* de uma PilhaJogada. O construtor de Jogo deve criar uma PilhaJogada (usando a função criaPilhaJogada, de PilhaJogada), e apontar este ponteiro para a nova estrutura

TAD Peca: um novo atributo deve ser acrescentado, para individualizar cada peça.

- int codPeca: código individual de uma peça. Cada peça no jogo deve ter seu próprio código. Considerando que temos 32 peças no jogo, uma sugestão seria numerar cada peça de 1 a 32

4. Novas Funcionalidades:

A. **Função "Desfazer" (*Undo*):**

Um jogador deve ter a possibilidade de se "arrepender" de uma jogada realizada. Se isso acontecer antes do adversário realizar uma nova jogada, o jogador arrependido poderá digitar "*undo*" e sua última jogada será desfeita e uma nova poderá ser realizada.

Esta função deve usar a PilhaJogada para recuperar a última jogada realizada. Os dados contidos no objeto da última jogada devem ser suficientes para permitir a reversão dessa jogada.

Considerações:

i. Toda jogada bem sucedida deve alocar um objeto Jogada, e armazenar suas informações nele

ii. Necessário apenas um nível de "*undo*". Se o comando "*undo*" for passado duas vezes seguidas, uma mensagem de erro informando "Operação não disponível" deve ser exibido

iii. O comando "*undo*" será digitado sempre na vez do adversário. Por isso, esse comando deve retornar a vez para o jogador anterior

Ex:

```
Brancas: "2d 4d"  <-- Jogada das brancas
Pretas: "undo"    <-- Na vez das pretas, as brancas se arrependem
Brancas: "2e 4e"  <-- Nova jogada das brancas
```

iv. A operação de "*undo*" deve desalocar o objeto Jogada que foi desfeito

B. Função "Salvar/Carregar":

Deve ser possível salvar uma partida em andamento e carregá-la mais tarde.

i. Os dados de uma partida devem ser salvos em um arquivo texto. O jogo deve dar opção ao jogador de nomear o arquivo para salvar o jogo, mas deve armazenar todos os arquivos em um diretório próprio

ii. É critério do grupo definir o formato dos dados salvos no arquivo. Entretanto, os dados salvos devem ser os seguintes:

- Situação de todas as peças (lista de peças)
- Situação das jogas realizadas (pilha de jogadas)

iii. O novo atributo "codPeca" deve ser usado para refazer corretamente as ligações entre Jogada e as peças (ex. cada lado possui 8 peões, mas cada Jogada precisa referenciar de forma correta o exato peão movido na jogada em questão)

iv. A função "criaJogo", no momento, cria todas as peças e as posiciona de forma estática. Isso deve ser alterado de forma que as peças sejam criadas de acordo com os dados presentes em um arquivo de jogo

v. Para tornar o funcionamento uniforme, devemos criar um arquivo de jogo "default.xdz", que não pode ser sobrescrito, contendo a situação inicial de um jogo

vi. A função "criaJogo" deve ser alterada para que pergunte ao jogador se ele deseja "Iniciar nova partida" ou "Carregar uma partida".

- A primeira opção deve usar "default.xdz" para iniciar um jogo novo.
- A segunda opção deve exibir uma lista de arquivos de jogo, e esperar que o usuário entre o nome de um deles, para então carregar os dados do arquivo escolhido

C. Função "Estatísticas":

O Jogo deve fornecer algumas estatísticas no final da partida (mesmo se o jogo for interrompido).

As estatísticas são as seguintes:

i. Total de jogadas Pretas

ii. Total de jogadas Brancas

iii. Jogada mais rápida do jogo

iv. Jogada mais demorada do jogo

v. As 3 jogadas mais rápidas das brancas

vi. As 3 jogadas mais rápidas das pretas

Para extrair as jogadas mais rápidas e mais lentas, a estratégia deve ser a seguinte:

i. A Pilha de Jogadas não é (e nem deve ser) ordenada pelo tempo da jogada
ii. Criamos um vetor de ponteiros para Jogada, com o tamanho da pilha (alocado dinamicamente)
iii. Varremos a pilha armazenando os dados no vetor
iv. Ordenamos o vetor usando algum método conhecido (preferencial: QuickSort)
v. Extraímos os dados para as estatísticas do vetor