Pontifícia Universidade Católica de Minas Gerais
Curso de Ciência da Computação
Disciplina: Algoritmos e Estruturas de Dados II

Trabalho Prático III

Regras Básicas

1. `extends Trabalho Prático 02`

2. Fique atento ao Charset dos arquivos de entrada e saída.

Harry Potter é uma série de sete romances de fantasia escrita pela autora britânica J.K. Rowling. A narrativa centraliza-se em Harry Potter, um jovem órfão que descobre, no seu décimo primeiro aniversário, que é um bruxo. Ele é convidado a estudar na Escola de Magia e Bruxaria de Hogwarts, onde aprende a prática da magia sob a orientação do gentil diretor Albus Dumbledore e de outros professores da escola. Harry descobre que é famoso no mundo dos bruxos por ter sobrevivido a um ataque letal do poderoso e maligno bruxo Lord Voldemort quando era apenas um bebê. Voldemort matou os pais de Harry, mas por algum motivo, sua maldição mortal não funcionou em Harry.

Ao longo dos livros, Harry é acompanhado por seus melhores amigos, Ronald Weasley e Hermione Granger. Juntos, eles enfrentam diversos desafios, incluindo o retorno de Voldemort, que busca não apenas conquistar o mundo bruxo, mas também destruir Harry, o único que pode impedir seus planos maléficos.

A série alcançou uma imensa popularidade, aclamação da crítica e sucesso comercial em todo o mundo. Além dos livros, a história de Harry Potter foi adaptada para uma série de filmes de sucesso, peças de teatro, jogos e uma vasta gama de produtos. Tornou-se um significativo fenômeno cultural e uma das séries de livros mais vendidas da história.

O arquivo **characters.csv** contém um conjunto de dados de personagens da série extraídos do site https://www.kaggle.com/. Essa base contém registros de 405 personagens. Este arquivo **sofreu algumas adaptações** para ser utilizado neste e nos próximos trabalhos práticos. Tal arquivo deve ser copiado para a pasta /tmp/. Quando reiniciamos o Linux, ele normalmente apaga os arquivos existentes na pasta /tmp/.

Estruturas Sequenciais

1. **Lista com Alocação Sequencial em Java**: Crie uma Lista de registros baseada na de inteiros vista na sala de aula. Sua lista deve conter todos os atributos e métodos existentes na lista de inteiros, contudo, adaptados para a classe `Personagem`. Lembre-se que, na verdade, temos uma lista de ponteiros (ou referências) e cada um deles aponta para um registo. Neste exercício, faremos inserções, remoções e mostraremos os elementos de nossa lista.

   Os métodos de inserir e remover devem operar conforme descrito a seguir, respeitando parâmetros e retornos. Primeiro, o `void inserirInicio(Personagem personagem)` insere um registro na primeira posição da Lista e remaneja os demais. Segundo, o `void inserir(Personagem personagem, int posição)` insere um registro na posição p da Lista, onde $p < n$ e n é o número de registros cadastrados. Em seguida, esse método remaneja os demais registros. O `void inserirFim(Personagem personagem)` insere um registro na última posição da Lista. O `Personagem removerInicio()` remove e retorna o primeiro registro cadastrado na Lista e remaneja os demais. O `Personagem remover(int posição)` remove e retorna o registro cadastrado na p-ésima posição da Lista e remaneja os demais. O `Personagem removerFim()` remove e retorna o último registro cadastrado na lista.

   A entrada padrão é composta por duas partes. A primeira é igual a entrada da primeira questão. As demais linhas correspondem a segunda parte. A primeira linha da segunda parte tem um número inteiro n indicando a quantidade de registros a serem inseridos/removidos. Nas próximas n linhas, tem-se n comandos de inserção/remoção a serem processados neste exercício. Cada uma dessas linhas tem uma palavra de comando: `II` inserir no início, `I*` inserir em qualquer posição, `IF` inserir no fim, `RI` remover no início, `R*` remover em qualquer posição e `RF` remover no fim. No caso dos comandos de inserir, temos também o nome do arquivo que contém o registro a ser inserido. No caso dos comandos de "em qualquer posição", temos também esse nome. No Inserir, a posição fica imediatamente após a palavra de comando. A saída padrão tem uma linha para cada registro removido sendo que essa informação será constituída pela palavra "(R)" e o atributo **name**. No final, a saída mostra os atributos relativos a cada registro cadastrado na lista após as operações de inserção e remoção.

2. **Lista com Alocação Sequencial em C**: Repita a anterior na linguagem C

3. **Pilha com Alocação Sequencial em Java**: Crie uma Pilha de registros baseada na pilha de inteiros vista na sala de aula. Neste exercício, faremos inserções, remoções e mostraremos os elementos de nossa pilha. A entrada e a saída padrão serão como as da questão anterior, contudo, teremos apenas os comandos `I` para inserir na pilha (empilhar) e `R` para remover (desempilhar).

4. **Fila Circular com Alocação Sequencial em C**: Crie uma classe *Fila Circular* de *Personagem*. Essa fila deve ter tamanho cinco. Em seguida, faça um programa que leia vários registros e insira seus atributos na fila. Quando o programa tiver que inserir um registro e a fila estiver cheia, antes, ele deve fazer uma remoção. A entrada padrão será igual à da questão anterior. A saída padrão será um número inteiro para cada registro inserido na fila. Esse número corresponde à média **arredondada do yearOfBirth** dos registros contidos na fila após cada inserção. Além disso, para cada registro removido da fila, a saída padrão também apresenta a palavra "(R)" e alguns atributos desse registro. Por último, a saída padrão mostra os registros existentes na fila seguindo o padrão da questão anterior.

Estruturas Flexíveis

5. **Lista com Alocação Flexível em Java**: Refazer a Questão 1 "Lista com Alocação Sequencial" usando lista dinâmica simples em Java.

6. **Pilha com Alocação Flexível em Java**: Refazer a Questão 3 "Pilha com Alocação Sequencial" em Java.

7. **Fila com Alocação Flexível em C**: Refazer a Questão 4 "Fila Circular com Alocação Sequencial" em C. Lembre-se que essa fila terá tamanho máximo igual a cinco.

8. **Quicksort com LISTA DINÂMICA DUPLAMENTE ENCADEADA em C**: Refaça a Questão "Quicksort" 10 do Trabalho Prático II - com lista duplamente encadeada. O nome do arquivo de log será matrícula_quicksort2.txt.

9. **Matriz Dinâmica em Java**: Complete o código da classe matriz dinâmica visto na sala de aula. A primeira tarefa consiste em, no construtor da classe Matriz, dados os números de linha e coluna, fazer as devidas alocações de células. As demais tarefas são as implementações dos métodos Matriz soma(Matriz), Matriz multiplicacao(Matriz), void mostrarDiagonalPrincipal() e void mostrarDiagonalSecundaria(). A entrada padrão é composta por vários casos de teste sendo que o número de casos é um inteiro contido na primeira linha da entrada. Em seguida, temos cada um dos casos de teste. Cada caso é composto por duas matrizes. Para cada caso de teste, temos que suas duas primeiras linhas contêm um número inteiro cada representando os números de linhas e de colunas da primeira matriz, respectivamente. Em seguida, temos os elementos da primeira matriz que estão representados nas próximas l linhas onde l é o número

de linhas dessa matriz. Cada uma dessas linhas têm c colunas onde c é o número de colunas dessa matriz. Nas duas linhas seguintes, temos os números de linhas e colunas da segunda matriz do caso de teste. As $l2$ linhas seguintes têm $c2$ colunas contendo os elementos da segunda matriz. $l2$ e $c2$ correspondem aos números de linhas e colunas da segunda matriz do caso de teste, respectivamente. A saída padrão contém várias linhas para cada caso de teste. As duas primeiras linhas de saída de um caso de teste correspondem às diagonais principal e secundaria da primeira matriz, respectivamente. As demais ls linhas de um caso de teste correspondem as linhas matriz obtida pela soma das duas matrizes do caso de teste sendo que essas linhas contêm cs colunas referentes às colunas da matriz de soma. Da mesma forma, as linhas seguintes do caso teste contêm lm linhas com cm colunas representando os elementos da matriz de multiplicação onde lm e cm são os números de linhas e colunas da matriz de multiplicação.

10. **Pilha com Alocação Flexível em C**: Refaça a questão 3 deste TP na linguagem C.

11. **Quicksort com LISTA DINÂMICA DUPLAMENTE ENCADEADA em Java**: Refaça a questão 8 deste TP na linguagem Java.