Pontifícia Universidade Católica de Minas Gerais
Curso de Ciência da Computação
Disciplina: Algoritmos e Estruturas de Dados II
Profs.: Felipe Domingos da Cunha, Max do Val Machado e Rodrigo Richard Gomes

Trabalho Prático II

Regras Básicas

1. extends Trabalho Prático 01

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

1 Estruturas Sequenciais

Star Wars é uma franquia do tipo space opera estadunidense criada pelo cineasta George Lucas que conta com uma série de oito filmes de fantasia científica e dois spin-offs. O primeiro filme foi lançado apenas com o título Star Wars em 25 de maio de 1977, e tornou-se um fenômeno mundial inesperado de cultura popular, sendo responsável pelo início da "era dos blockbusters": Super produções cinematográficas que fazem sucesso nas bilheterias e viram franquias com brinquedos, jogos, livros, etc. Foi seguido por duas sequências, The Empire Strikes Back e Return of the Jedi, lançadas com intervalos de três anos, formando a trilogia original. Esta primeira trilogia segue o trio icônico: Luke Skywalker, Han Solo e Princesa Leia, que luta na Aliança Rebelde para derrubar o tirano Império Galáctico; paralelamente ocorre a jornada de Luke para se tornar um cavaleiro Jedi e a luta contra Darth Vader, um ex-Jedi que sucumbiu ao Lado Sombrio da Força e ao Imperador.

Depois de 16 anos sem filmes novos lançados, uma nova trilogia chamada de prequela começou em 1999 com The Phantom Menace. Esta volta no tempo para contar como Anakin Skywalker se transformou em Darth Vader e acompanha a queda da Ordem Jedi e da República Galáctica substituída pelo Império. Sendo também lançada com intervalos de três anos, com o último lançado em 2005. As reações à trilogia original foram extremamente positivas, enquanto a trilogia prequela recebeu reações mistas tanto da crítica especializada como do público. Mesmo assim, todos os filmes foram bem sucedidos nas bilheterias e receberam indicações ou ganharam prêmios no Óscar.

O arquivo **personagens.zip** contém um conjunto de dados de personagens da franquia Star Wars, oriundo de respostas para requisições que foram feitas a API `swapi.co/` em 01/02/2020. Estes arquivos **sofreram 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/.

Implemente os itens pedidos a seguir.

1. **Classe em Java**: Crie uma classe *Personagem* seguindo todas as regras apresentadas no slide unidade01g_conceitosBasicos_introducaoOO.pdf. Sua classe terá os atributos privados nome (String), altura (int), peso (double), corDoCabelo (String), codDaPele (String), corDosOlhos (String), anoNascimento (String), genero (String), homeworld (String). Sua classe também terá pelo menos dois construtores, e os métodos *gets*, *sets*, *clone*, *imprimir* e *ler*. O método *imprimir* mostra os atributos do registro (ver cada linha da saída padrão) e o *ler* lê os atributos de um registro.

   A entrada padrão é composta por várias linhas e cada uma contém uma string indicando o arquivo a ser lido. A última linha da entrada contém a palavra FIM. A saída padrão também contém várias linhas, uma para cada registro contido em uma linha da entrada padrão.

2. **Registro em C**: Repita a anterior criando o registro Personagem na linguagem C.

3. **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 **nome**. No final, a saída mostra os atributos relativos a cada registro cadastrado na lista após as operações de inserção e remoção.

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

5. **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).

6. **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 da altura** 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.

2 Pesquisa e Ordenação

7. **Pesquisa Sequencial em Java**: Faça a inserção de alguns registros no final de uma Lista e, em seguida, faça algumas pesquisas sequenciais. A chave primária de pesquisa será o atributo **nome**. A entrada padrão é composta por duas partes onde a primeira é igual a entrada da primeira questão. As demais linhas correspondem a segunda parte. A segunda parte é composta por várias linhas. Cada uma possui um elemento que deve ser pesquisado na Lista. A última linha terá a palavra FIM. A saída padrão será composta por várias linhas contendo as palavras SIM/NAO para indicar se existe cada um dos elementos pesquisados. Além disso, crie um arquivo de log na pasta corrente com o nome matrícula_sequencial.txt com uma única linha contendo sua matrícula, tempo de execução do seu algoritmo e número de comparações. Todas as informações do arquivo de log devem ser separadas por uma tabulação '\t'.

8. **Pesquisa Binária em C**: Repita a questão anterior, contudo, usando a Pesquisa Binária. A entrada e a saída padrão serão iguais as da questão anterior. O nome do arquivo de log será matrícula_binaria.txt. A entrada desta questão está ordenada.

9. **Ordenação por Seleção em Java**: Na classe Lista, implemente o algoritmo de ordenação por seleção considerando que a chave de pesquisa é o atributo **nome**. A entrada e a saída padrão são iguais as da primeira questão, contudo, a saída corresponde aos registros ordenados. Além disso, crie um arquivo de log na pasta corrente com o nome matrícula_selecao.txt com uma única linha contendo sua matrícula, número de comparações (entre elementos do *array*), número de movimentações (entre elementos do *array*) e o tempo de execução do algoritmo de ordenação. Todas as informações do arquivo de log devem ser separadas por uma tabulação '\t'.

10. **Ordenação por Seleção Recursiva em C**: Repita a questão anterior, contudo, usando a Seleção Recursiva. A entrada e a saída padrão serão iguais as da questão anterior. O nome do arquivo de log será matrícula_selecaoRecursiva.txt.

11. **Ordenação por Inserção em Java**: Repita a questão de Ordenação por Seleção, contudo, usando o algoritmo de Inserção, fazendo com que a chave de pesquisa seja o atributo **anoNascimento**. O nome do arquivo de log será matrícula_insercao.txt.

12. **Shellsort em C**: Repita a questão de Ordenação por Seleção, contudo, usando o algoritmo Shellsort, fazendo com que a chave de pesquisa seja o atributo **peso**. O nome do arquivo de log será matrícula_shellsort.txt.

13. **Heapsort em Java**: Repita a questão de Ordenação por Seleção, contudo, usando o algoritmo Heapsort, fazendo com que a chave de pesquisa seja o atributo **altura**. O nome do arquivo de log será matrícula_heapsort.txt.

14. **Quicksort em C**: Repita a questão de Ordenação por Seleção, contudo, usando o algoritmo Quicksort, fazendo com que a chave de pesquisa seja o atributo **corDoCabelo**. O nome do arquivo de log será matrícula_quicksort.txt.

15. **Counting Sort em Java**: Repita a questão de Ordenação por Seleção, contudo, usando o algoritmo Counting Sort, fazendo com que a chave de pesquisa seja o atributo **peso**. O nome do arquivo de log será matrícula_countingsort.txt.

16. **Bolha em C**: Repita a questão de Ordenação por Seleção, contudo, usando o algoritmo da Bolha, fazendo com que a chave de pesquisa seja o atributo **anoNascimento**. O nome do arquivo de log será matrícula_bolha.txt.

17. **Mergesort em Java**: Repita a questão de Ordenação por Seleção, contudo, usando o algoritmo Mergesort, fazendo com que a chave de pesquisa seja o atributo **homeWorld**. O nome do arquivo

de log será matrícula_mergesort.txt.

18. **Radixsort em C**: Repita a questão de Ordenação por Seleção, contudo, usando o algoritmo Radixsort, fazendo com que a chave de pesquisa seja o atributo **homeWorld**. O nome do arquivo de log será matrícula_radixsort.txt.

19. **Ordenação PARCIAL por Seleção em Java**: Refaça a Questão "Ordenação por Seleção" considerando a ordenação parcial com k igual a 10.

20. **Ordenação PARCIAL por Inserção em C**: Refaça a Questão "Ordenação por Inserção" considerando a ordenação parcial com k igual a 10.

21. **Heapsort PARCIAL em C**: Refaça a Questão "Heapsort" considerando a ordenação parcial com k igual a 10.

22. **Quicksort PARCIAL em Java**: Refaça a Questão "Quicksort" considerando a ordenação parcial com k igual a 10.