|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | **PREENCHA OS CAMPOS EM AMARELO** | **PROVA P3** | |
|  |
| Disc.: | **ECM306 – tópicos avançados em estrutura de dados** | |
| Curso: | **ENGENHARIA DA COMPUTAÇÃO** | |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Aluno: | | **Pedro Wilian Palumbo Bevilacqua** | | | | | | | | | Test Code:  **MFVYS** |
| Curso: | **ECM** | | | Série | **3ª** | Período: | **Matutino** |  |  | |
| São Caetano do Sul, 25 de setembro de 2025. | | | | | | | | | RA: | **23-01307.9** |
| Assinatura: | | | **Pedro Wilian** | | | | | | Nota: |  |

**Instruções da Prova**

1. Esta prova é individual e prática, devendo ser realizada nos computadores do IMT, sendo permitido ao aluno, se assim desejar, utilizar seu próprio computador, sob sua inteira responsabilidade;
2. Não poderá haver acesso à Internet, sob nenhuma circunstância, exceto ao CANVAS LMS do próprio aluno e, mesmo assim, somente em duas etapas: para receber (“baixar”) as questões de sua prova na máquina em uso; e para entregar (“subir”) as resoluções das questões de sua prova no devido local de entrega, na mesma plataforma.

**IMPORTANTE**: O aluno deverá informar ao professor quando fará os dois acessos permitidos a ele ao CANVAS LMS, pela Internet;

1. Poderá haver consulta a qualquer material do próprio aluno, seja ele físico (livros, artigos, material de aula etc.) ou virtual (livros, artigos, material de aula, exercícios, resoluções etc.), desde que esse material esteja previamente armazenado em seu computador ou em qualquer dispositivo de armazenamento externo (*pendrive*, *hd* externo etc.).

**IMPORTANTE**: Não será permitido o acesso pela Internet a pastas compartilhadas (*Google* *Drive*, *OneDrive* etc.), nem a repositórios virtuais (*GitHub*, *GitLab*, *BitBucket* etc.), mesmo sendo de posse e administração do próprio aluno;

1. O aluno deverá responder às questões da prova no próprio arquivo da prova (.*docx*) devidamente identificado (RA e Nome completo do Aluno) e quando for o caso, gerando os códigos necessários, somente na linguagem de programação JAVA e utilizando sempre e somente o paradigma da Programação Orientada à Objetos visto nas aulas;
2. Para realizar a entrega da prova na plataforma CANVAS LMS, o aluno deverá gerar e entregar um único arquivo compactado (*.rar* ou .*zip*), tendo seu RA e seu NOME completo como nome desse arquivo. Nesse arquivo compactado o aluno deverá fornecer, obrigatoriamente, os seus dados acadêmicos preenchidos na prova, bem como a resolução das questões nela solicitadas (.*docx*), além dos arquivos e códigos gerados em suas resoluções, uma pasta para cada questão, contendo as suas classes e demais arquivos que possibilitem sua posterior execução pelo professor, durante a resolução.

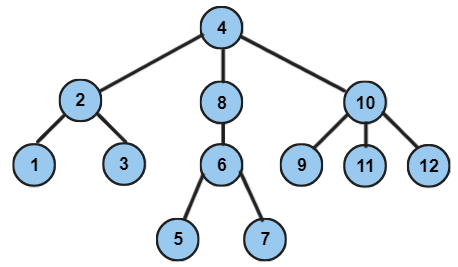
**IMPORTANTE**: Na correção, para executar o código gerado pelo aluno em sua prova, o professor seguirá exatamente as instruções fornecidas pelos alunos e contidas nas resoluções das provas! Caso não obtenha sucesso, a questão será considerada errada.

1. Não poderá haver troca de informações, nem de materiais, sejam físicos ou virtuais, entre os alunos durante a prova;
2. Não é permitido ao estudante se ausentar da sala antes da entrega da prova;
3. Celulares e outros equipamentos eletrônicos (exceto o notebook do aluno, se assim optar) devem permanecer desligados enquanto o estudante estiver na sala;
4. O tempo limite para realização da prova é de **90** minutos;
5. Mantenha sobre a carteira apenas um documento com foto, caneta, lápis e borracha;
6. O entendimento das questões faz parte da avaliação;
7. O tempo mínimo de permanência na sala é de **30** minutos;
8. O estudante que chegar atrasado em até **30** minutos do início da prova poderá fazê-la.

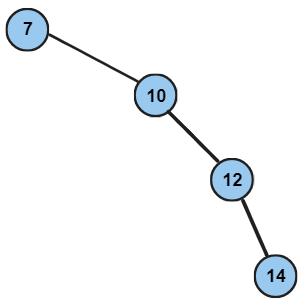
**Afirmações de 1 à 3**:

Dadas as figuras 1, 2 e 3 abaixo, verifique se cada **Afirmação de 1 à 3** é *Verdadeira* ou *Falsa* e depois responda à **Questão 1**:

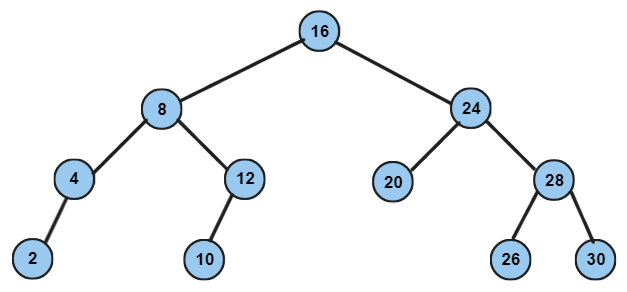
FIGURAS:



*Figura 3*



*Figura 2*

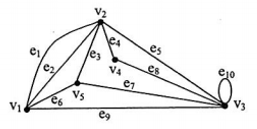


*Figura 1*

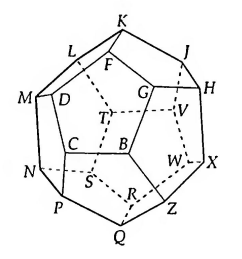
**Afirmações de 4 à 9**:

Dadas as figuras 4, 5 e 6 abaixo, verifique se cada **Afirmação de 4 à 9** é *Verdadeira* ou *Falsa* e depois responda às **Questões 2 e 3**:

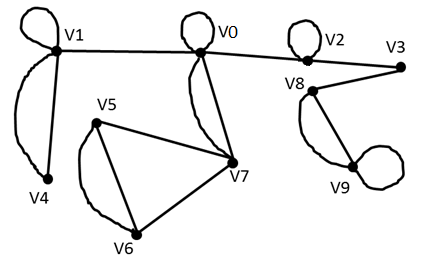
FIGURAS:



*Figura 6*



*Figura 5*



*Figura 4*

AFIRMAÇÕES:

A1 – Sendo a Estrutura de Dados da figura 1 uma Árvore Binária, de acordo com a implementação vista em aula, pode-se afirmar que o nó *16* foi o primeiro a ser inserido nessa estrutura.

A2 – Sendo a Estrutura de Dados da figura 2 uma Árvore Binária, de acordo com a implementação vista em aula, pode-se afirmar que ela está balanceada.

A3 – Sendo a Estrutura de Dados da figura 3 uma Árvore, de acordo com a implementação vista em aula, pode-se afirmar que o nó *12* tem o *10* como nó “pai” e o *9* como nó “irmão mais velho”.

A4 – Dado o Grafo *G = (V, E)* representado pela a figura 4, pode-se afirmar que sua quantidade de arestas é igual à *16*.

A5 – Dado o Grafo *G = (V, E)* representado pela a figura 4, pode-se afirmar que a soma de todos os graus de seus nós é igual à *32*.

A6 – Dado o Grafo *G = (V, E)* representado pela a figura 4, pode-se afirmar que ele possui *3* vértices ímpares.

A7 – Dado o Grafo *G = (V, E)* representado pela a figura 5, pode-se afirmar que *G* é um Grafo Não Euleriano.

A8 – Dado o Grafo *G = (V, E)* representado pela a figura 6 e *W* = *v1 v5 v2 v4 v3 v5 v2*, pode-se afirmar que *W* é um *CIRCUITO*.

A9 – Dado o Grafo *G = (V, E)* representado pela a figura 6 e *W* = *v1 v5 v2 v4 v3 v5 v2*, pode-se afirmar que *W* **não** é um *CAMINHO*.

**QUESTÕES 1, 2 e 3:** (1 *ponto* cada)

Q1 – Assinale a única alternativa correta:

a) As afirmações A1, A2 e A3 são verdadeiras;

b) Apenas a afirmação A1 é falsa, dentre A1, A2 e A3;

c) As afirmações A1, A2 e A3 são falsas;

d) Apenas a afirmação A3 é verdadeira, dentre A1, A2 e A3;

e) N.D.A.

Q2 – Assinale a única alternativa correta:

a) As afirmações A4, A5 e A6 são verdadeiras;

b) Apenas a afirmação A5 é verdadeira, dentre A4, A5 e A6;

c) As afirmações A4, A5 e A6 são falsas;

d) Apenas a afirmação A6 é falsa, dentre A4, A5 e A6;

e) N.D.A.

Q3 – Assinale a única alternativa correta:

a) As afirmações A7, A8 e A9 são verdadeiras;

b) Apenas a afirmação A8 é verdadeira, dentre A7, A8 e A9;

c) As afirmações A7, A8 e A9 são falsas;

d) Apenas a afirmação A8 é falsa, dentre A7, A8 e A9;

e) N.D.A.

**QUESTÃO 4:** (4 *pontos*)

Gerado por qualquer *IDE*, fornecer o código fonte, em *Java*, das classes necessárias (incluindo a de execução), implementadas sob o paradigma da *Programação Orientada a Objetos*, além do programa ***.jar*** que as executa diretamente, objetivando resolver o problema descrito a seguir.

ATENÇÃO:

1. Se o programa ***.jar*** fornecido não executar automaticamente a aplicação desenvolvida, a nota da questão será **decrementada** de ***1*** (***um***) ***ponto***;
2. As classes soluções desta questão (arquivos ***.java***) deverão ser compactadas em um único arquivo (**.*zip*** ou **.*rar***) em conjunto com o seu respectivo arquivo ***.jar*** funcional, além deste arquivo ***.docx*** da prova, contendo todas as respostas para as outras questões e a devida identificação do aluno e entregue em resposta à tarefa do *CANVAS LMS* da prova.

Problema:

Utilizando-se apenas os códigos vistos e praticados em aula e os códigos desenvolvidos pelo próprio aluno para esta disciplina, através de uma aplicação implementada durante a prova, deseja-se criar uma **Estrutura de Dados – ED** dinâmica, baseada em **Fila de Prioridades (*Heap*)**.

A aplicação deverá dar suporte ao atendimento de um centro de saúde, onde os pacientes serão chamados por ordem de chegada, através da geração de senha e posterior chamada.

Devido às Leis Federais, será necessário impor a essa aplicação uma Fila de Prioridades, sobrepondo às senhas por ordem de chegada, devendo haver 5 (cinco) categorias, a saber:

Categoria Prioriadade Descrição

1 Máxima Idosos com 80 anos ou mais

2 Alta Idosos com 60 anos ou mais

3 Média Grávidas, Puérperas e Deficientes

4 Baixa Obesos e Pessoas com acompanhadas de crianças

5 Mínima Sem restrições

A aplicação deverá cumprir os seguintes passos, em sequência:

* + 1. O paciente informa à aplicação, na chegada, através da digitação (*1*, *2*, *3*, *4* ou *5*), a sua categoria;
    2. A aplicação informa ao paciente, através da tela, qual é a sua senha (ex.: *1001*), de acordo com sua categoria e ordem de chegada;
    3. O passo (ii) se repete a qualquer momento que um novo paciente informar à aplicação a sua categoria;
    4. O atendente solicita à aplicação, através da digitação (*?*), a chamada do próximo paciente pela respectiva senha, armazenada na fila prioridades, através da sua apresentação em tela;
    5. O passo (iv) se repete a qualquer momento que o atendente estiver livre e enquanto houver no mínimo uma senha na fila de prioridades;
    6. A aplicação apresenta em tela a mensagem “Fila Vazia” quando não houver mais senhas armazenadas.

Casos de Teste:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Ator | Missão | Ação (teclado) | Reação (tela) | ! |
| 01 | Atendente | Iniciar a aplicação | Executa *app.jar* | Mostra “Sistema ACME” |  |
| 02 | Paciente #1 | Informar categoria 5 | Digita ‘5’ | Mostra “Senha 5001” |  |
| 03 | Paciente #2 | Informar categoria 4 | Digita ‘4’ | Mostra “Senha 4001” |  |
| 04 | Paciente #3 | Informar categoria 3 | Digita ‘3’ | Mostra “Senha 3001” |  |
| 05 | Paciente #4 | Informar categoria 3 | Digita ‘3’ | Mostra “Senha 3002” |  |
| 06 | Paciente #5 | Informar categoria 2 | Digita ‘2’ | Mostra “Senha 2001” |  |
| 07 | Paciente #6 | Informar categoria 1 | Digita ‘1’ | Mostra “Senha 1001” |  |
| 08 | Paciente #7 | Informar categoria 2 | Digita ‘2’ | Mostra “Senha 2002” |  |
| 09 | Atendente | Chamar próximo | Digita ‘?’ | Mostra “Paciente 1001” |  |
| 10 | Atendente | Chamar próximo | Digita ‘?’ | Mostra “Paciente 2001” |  |
| 11 | Paciente #8 | Informar categoria 1 | Digita ‘1’ | Mostra “Senha 1002” |  |
| 12 | Paciente #9 | Informar categoria 5 | Digita ‘5’ | Mostra “Senha 5002” |  |
| 13 | Atendente | Chamar próximo | Digita ‘?’ | Mostra “Paciente 1002” |  |
| 14 | Atendente | Chamar próximo | Digita ‘?’ | Mostra “Paciente 2002” |  |
| 15 | Atendente | Chamar próximo | Digita ‘?’ | Mostra “Paciente 3001” |  |
| 16 | Atendente | Chamar próximo | Digita ‘?’ | Mostra “Paciente 3002” |  |
| 17 | Atendente | Chamar próximo | Digita ‘?’ | Mostra “Paciente 4001” |  |
| 18 | Atendente | Chamar próximo | Digita ‘?’ | Mostra “Paciente 5001” |  |
| 19 | Atendente | Chamar próximo | Digita ‘?’ | Mostra “Paciente 5002” |  |
| 20 | Atendente | Chamar próximo | Digita ‘?’ | Mostra “Fila Vazia” |  |

**QUESTÃO 5:** (3 *pontos*)

Gerado por qualquer *IDE*, fornecer o código fonte, em *Java*, das classes necessárias (incluindo a de execução), implementadas sob o paradigma da *Programação Orientada a Objetos*, além do programa ***.jar*** que as executa diretamente, objetivando resolver o problema descrito a seguir.

ATENÇÃO:

1. Se o programa ***.jar*** fornecido não executar automaticamente a aplicação desenvolvida, a nota da questão será **decrementada** de ***1*** (***um***) ***ponto***;
2. As classes soluções desta questão (arquivos ***.java***) deverão ser compactadas em um único arquivo (**.*zip*** ou **.*rar***) em conjunto com o seu respectivo arquivo ***.jar*** funcional, além deste arquivo ***.docx*** da prova, contendo todas as respostas para as outras questões e a devida identificação do aluno e entregue em resposta à tarefa do *CANVAS LMS* da prova.

Problema:

Utilizando-se apenas os códigos vistos e praticados em aula e os códigos desenvolvidos pelo próprio aluno para esta disciplina, através de uma aplicação implementada durante a prova, deseja-se criar uma **Estrutura de Dados – ED** dinâmica, baseada em **Árvores**, capaz de armazenar um caractere (***char***) por nó, representando a inicial (a ser digitada) de um nome.

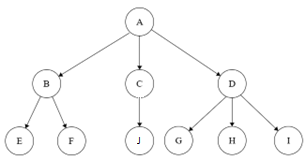
Essa **ED** deverá nascer **vazia** e a ela serão informados, a cada solicitação do usuário (via teclado), o **primeiro caractere** do **nome** do **novo nó filho** em questão e o **primeiro caractere** do **nome** do **pai desse nó filho**, um após o outro, sem repetições de caractere.

A **ED** deverá inserir O novo nó filho abaixo do seu nó pai, no local destinado a ele, seguindo as respectivas regras para a inserção de um **novo** **nó** em uma Árvore.

Após cada inserção e sua a respectiva alocação, a **ED** deverá ter o conteúdo de todos os seus nós **impressos** em tela, em sequência, através de um percurso em **PRÉ-ORDEM**.

A aplicação apresentada deverá funcionar como uma Árvore de forma correta para quaisquer valores de entrada, em qualquer ordem de inserção!

Casos de Teste:



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | Missão | Ação 1: Nó (teclado) | Ação 2: Pai (teclado) | Reação 1: Tamanho (tela) | Reação 2:  Disposição (tela) | ! |
| 00 | Iniciar aplicação | Executa *app.jar* | N/A | 0 | “Árvore Vazia” | X |
| 01 | Inserir nó *A*  na *raiz* | ‘A’ | *null* | 1 | “A” |  |
| 02 | Inserir nó *B*  com o pai *A* | ‘B’ | ‘A’ | 2 | “A, B” |  |
| 03 | Inserir nó *C*  com o pai *A* | ‘C’ | ‘A’ | 3 | “A, B, C” |  |
| 04 | Inserir nó *D*  com o pai *A* | ‘D’ | ‘A’ | 4 | “A, B, C, D” |  |
| 05 | Inserir nó *E*  com o pai *B* | ‘E’ | ‘B’ | 5 | “A, B, E, C, D” |  |
| 06 | Inserir nó *F*  com o pai *B* | ‘F’ | ‘B’ | 6 | “A, B, E, F, C, D” |  |
| 07 | Inserir nó *G*  com o pai *D* | ‘G’ | ‘D’ | 7 | “A, B, E, F, C, D, G” |  |
| 08 | Inserir nó *H*  com o pai *D* | ‘H’ | ‘D’ | 8 | “A, B, E, F, C, D, G, H” |  |
| 09 | Inserir nó *I*  com o pai *D* | ‘I’ | ‘D’ | 9 | “A, B, E, F, C, D, G, H, I” |  |
| 10 | Inserir nó *J*  com o pai *C* | ‘J’ | ‘C’ | 10 | “A, B, E, F, C, J, D, G, H, I” |  |