Prova III - Teoria

Entrega 28 jun em 10:40 **Pontos** 11 **Perguntas** 32
Disponível 28 jun em 8:40 - 28 jun em 10:40 aproximadamente 2 horas
Limite de tempo 120 Minutos

Instruções

Esta prova TEÓRICA vale 10 pontos. Ela tem 30 questões de verdadeiro ou falso e duas questões abertas de código. As questões de verdadeiro ou falso valem 0,2 e as abertas, 2.5. O somatório dessas notas é igual a 11. Fizemos isso porque, novamente, NÃO será permitido retornar nas questões: nem fechadas, nem abertas. A submissão das questões abertas será efetuada em campo de arquivo texto. Adotaremos novamente o recurso do vídeo explicando a solução. Nesse caso, o vídeo será postado no YouTube e o aluno submeterá no Canvas apenas o link desse vídeo. A submissão do vídeo poderá ser efetuada até às 23:59 do dia da prova. Anularemos os vídeos que tiverem data de postagem posterior ao permitido.

A prova teórica da manhã acontecerá de 8:50 às 10:30 e a da tarde, de 13:30 às 15:10. A prova será aberta 10 minutos mais cedo e fechará 10 minutos mais tarde.

Este teste foi travado 28 jun em 10:40.

Histórico de tentativas

	Tentativa	Tempo	Pontuação
MAIS RECENTE	Tentativa 1	108 minutos	10,8 de 11

ⓘ As respostas corretas não estão mais disponíveis.

Pontuação deste teste: **10,8** de 11
Enviado 28 jun em 10:28
Esta tentativa levou 108 minutos.

Pergunta 1 — 0,2 / 0,2 pts

Considere uma tabela *hash* com *rehash* contendo cinco posições e usando as funções de transformação apresentadas abaixo, podemos afirmar que após a inserção dos elementos 1, 6, 5, 10, 3 e 4, a tabela resultante será [5 1 6 3 4].

```
int hash(int key){
    return key%5;
}

int rehash(int key){
    return ++key%5;
}
```

- (●) Verdadeiro
- ( ) Falso

A afirmação é verdadeira, pois o 10 não será inserido.

Pergunta 2

0,2 / 0,2 pts

No método de transformação (*hashing*), os registros armazenados em uma tabela são diretamente endereçados a partir de uma transformação aritmética sobre a chave de pesquisa. Com relação às funções de transformação e colisões, podemos afirmar que os métodos de transformação mais conhecidos e funcionais incluem o resto da multiplicação (TRE-PI'16, adaptado).

- (●) Falso
- ( ) Verdadeiro

A afirmação é falsa. Os métodos mais conhecidos utilizam o resto da divisão inteira.

Pergunta 3

0,2 / 0,2 pts

No método de transformação (*hashing*), os registros armazenados em uma tabela são diretamente endereçados a partir de uma transformação aritmética sobre a chave de pesquisa. Com relação às funções de transformação e colisões, podemos afirmar que a técnica de endereçamento aberto (*hash* direto) resolve colisões usando uma matriz esparsa (TRE-PI'16, adaptado).

- (•) Falso
- ( ) Verdadeiro

A afirmação é falsa. O *hash* direto consiste em inserir os elementos que sofreram colisões dentro da própria tabela. Por exemplo, podemos utilizar área de reserva ou *overflow* e funções de *rehash*.

Pergunta 4

0,2 / 0,2 pts

Uma estrutura de dados eficiente é a tabela *hash*. Um desafio no projeto dessa estrutura é seu dimensionamento. Em uma tabela adequadamente dimensionada, com *n* chaves, o número médio de acessos para localização de uma chave é $\Theta(1)$ comparações (TJ-PI'15, adaptada).

- ( ) Falso
- (•) Verdadeiro

Uma tabela *hash* é bem dimensionada quando seu custo é constante, $\Theta(1)$ comparações.

Pergunta 5

0,2 / 0,2 pts

No método de transformação (*hashing*), os registros armazenados em uma tabela são diretamente endereçados a partir de uma transformação aritmética sobre a chave de pesquisa. Com relação às funções de transformação e colisões, podemos afirmar que a técnica de endereçamento separado (*hash* indireto) resolve colisões usando uma estrutura auxiliar externa (e.g., lista linear encadeada) para cada endereço da tabela (TRE-PI'16, adaptado).

- (●) Verdadeiro
- ( ) Falso

A afirmação é verdadeira e ela descreve a técnica de *hash* indireto.

Pergunta 6

0,2 / 0,2 pts

Ao fragmentarmos um nó do tipo 4, a altura da árvore aumenta em um nível.

- (●) Falso
- ( ) Verdadeiro

A afirmação é falsa. Nas fragmentações, o elemento do meio "sobe" para seu pai sem aumentar o nível da árvore. A única exceção acontece quando temos a fragmentação da raiz, pois, como não existe um pai, esse é criado.

Incorreta

Pergunta 7

0 / 0,2 pts

A inserção dos números 4, 24, 10, 16, 2, 22, 18, 14, 8, 26, 20, 12 e 6 em uma 2.3.4 usando as técnicas de fragmentação por ascenção e na descida gera a mesma árvore resultante.

- Verdadeiro
- (•) Falso

A afirmação é verdadeira. A inserção usando as duas técnicas é igual até a inserção do 20. Elas ficam diferentes após a inserção do 12 e voltam a ficar iguais com a inserção do 6.

Pergunta 8

0,2 / 0,2 pts

Uma vantagem da fragmentação na descida é que, na ascensão é preciso uma pilha para restaurar o equilíbrio da árvore repassando o caminho inverso de pesquisa no caso de fragmentações.

- Falso
- (•) Verdadeiro

A afirmação é verdadeira. A fragmentação na descida faz apenas atualizações locais envolvendo um nó e seu pai. Por outro lado, a técnica por ascensão faz com que exista um efeito cascata de fragmentações.

Pergunta 9

0,2 / 0,2 pts

Na árvore 2.3.4, após uma inserção utilizando fragmentação na descida, podemos garantir a inexistência de nós do tipo 4 no caminho entre a raiz e a folha em que aconteceu a inserção.

- (●) Falso
- ( ) Verdadeiro

A afirmação é falsa. Na inserção com fragmentação na descida, quando chegamos em um nó do tipo 4, esse é fragmentado. Nessa fragmentação, se o pai era do tipo 3, ele ficará sendo do tipo 4. Da mesma forma, se inserirmos em uma folha do tipo 3, essa ficará sendo do tipo 4.

Pergunta 10

0,2 / 0,2 pts

A técnica de inserção com fragmentação na descida na 2.3.4 pode aumentar, reduzir ou manter o número de nós do tipo 4.

- ( ) Falso
- (●) Verdadeiro

A afirmação é verdadeira. Na inserção com fragmentação na descida, quando temos um nó do tipo 4 esse será fragmentado o que pode levar a uma redução do número de nós do tipo 4. Por outro lado, quando inserimos o elemento em uma folha do tipo 3, essa vira do tipo 4, aumentando o número de nós desse tipo. Temos ainda que durante a inserção podemos não ter fragmentações, mantendo o número de nós do tipo 4 existentes na árvore. Esse número também é mantido quando para cada Fragmentação, temos um nó do tipo 3 que vira do tipo 4.

Pergunta 11

0,2 / 0,2 pts

Uma das estruturas de dados utilizadas na modelagem de sistemas de software denomina-se árvore rubro-negra. Nesse caso, um nó será vermelho quando, na árvore 2-3-4 correspondente, o valor desse nó for gêmeo do nó pai. Caso contrário, o nó é preto. Em uma árvore rubro-negra temos que se um nó é preto, seus filhos são pretos.

- (●) Falso
- ( ) Verdadeiro

A afirmação é falsa conforme os conceitos de árvores alvinegras apresentados na sala de aula.

Pergunta 12

0,2 / 0,2 pts

Em uma árvore alvinegra, se tivermos dois nós pretos seguidos, temos que fazer uma rotação com o avô.

- (●) Verdadeiro
- ( ) Falso

A afirmação é verdadeira conforme os conceitos de árvores alvinegras apresentados na sala de aula.

Incorreta

Pergunta 13

0,2 / 0,2 pts

Uma árvore rubro-negra possui 18 valores inteiros distintos armazenados em seus 18 nós. A função recursiva boolean busca (int val) visita os nós desse tipo de árvore à procura de um determinado valor (val). O algoritmo utilizado tira partido das características de uma árvore rubro-negra, com o objetivo de ser o mais eficiente possível. Podemos afirmar que o número máximo de chamadas da função que será necessário para informar se um determinado valor está, ou não, armazenado na árvore é igual a seis (BNDES'13, adaptada).

- Verdadeiro
- Falso

A afirmação é verdadeira e pode ser confirmada executando o código disponibilizado pelo professor.

Pergunta 14

0,2 / 0,2 pts

Uma das estruturas de dados utilizadas na modelagem de sistemas de software denomina-se árvore rubro-negra. Nesse caso, um nó será vermelho quando, na árvore 2-3-4 correspondente, o valor desse nó for gêmeo do nó pai. Caso contrário, o nó é preto. Em uma árvore rubro-negra temos que a quantidade de nós vermelhos é sempre igual à de pretos.

- Verdadeiro
- Falso

A afirmação é falsa conforme os conceitos de árvores alvinegras apresentados na sala de aula.

Pergunta 15

0,2 / 0,2 pts

Em uma árvore alvinegra, quando um nó tem 2 filhos pretos, fazemos a inversão das cores entre os filhos e o pai.

- (●) Verdadeiro
- ( ) Falso

A afirmação é verdadeira conforme os conceitos de árvores alvinegras apresentados na sala de aula.

Pergunta 16

0,2 / 0,2 pts

Todos os n nomes de uma lista de assinantes de uma companhia telefônica foram inseridos, em ordem alfabética, em três estruturas de dados: uma árvore binária de busca, uma árvore AVL e uma árvore bicolor. As alturas resultantes das três árvores são $\Theta(n)$, $\Theta(lg(n))$ e $\Theta(lg(n))$, respectivamente (PETROBRAS'12, adaptada)

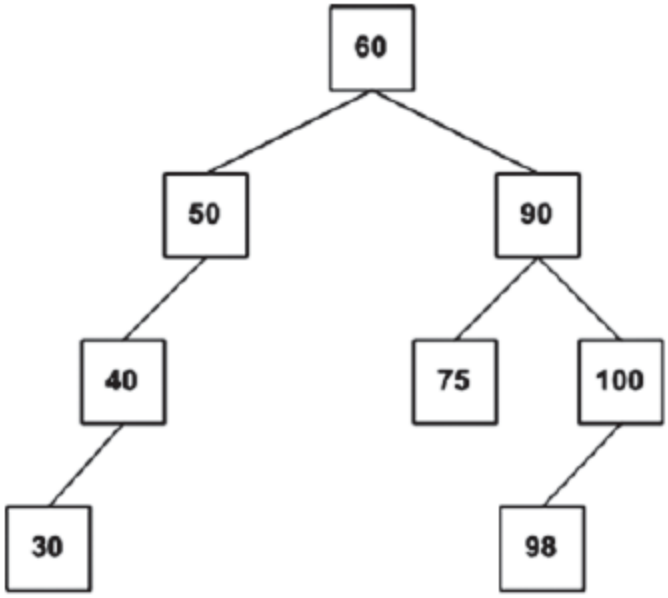
- (●) Verdadeiro
- ( ) Falso

A afirmação é verdadeira. A AB não é balanceada e as demais são.

Pergunta 17

0,2 / 0,2 pts

A estrutura de dados AVL é uma árvore binária de busca balanceada criada pelos soviéticos Adelson, Velsky e Landis em 1962. Podemos afirmar que a figura abaixo representa uma árvore AVL (PETROBRAS'12, adaptada).



- Verdadeiro
- Falso

A afirmação é falsa conforme o conceito de árvore AVL.

Pergunta 18

0,2 / 0,2 pts

A AVL é uma árvore de busca autobalanceada. Isso significa que as alturas das duas sub-árvores a partir de cada nó são exatamente iguais (SEFAZ-PI'15, adaptado).

- (●) Falso
- ( ) Verdadeiro

A afirmação é falsa conforme o conceito de árvore AVL.

Pergunta 19

0,2 / 0,2 pts

A AVL é uma árvore de busca autobalanceada. Isso significa que as alturas das duas sub-árvores a partir de cada nó diferem no máximo em duas unidades (SEFAZ-PI'15, adaptado).

- ( ) Verdadeiro
- (●) Falso

A afirmação é falsa conforme o conceito de árvore AVL.

Pergunta 20

0,2 / 0,2 pts

Avaliando os códigos abaixo nas linguagens C e C++, é correto afirmar que os dois imprimem os mesmos valores para as variáveis "x" e "y" no final de suas execuções. Isso acontece mesmo com a passagem de parâmetros do primeiro código sendo por valor e a do segundo, sendo por referência.

Código em C:

```
void troca(double *x, double *y){
        double aux = *x;
        *x = *y;
```

```
        *y = aux;
}

int main(){
        double x = 7.0;
        double y = 5.0;
        troca(&x, &y);
        printf("X: %lf Y: %lf", x, y);
}
```

Código em C++:

```
void troca(double& x, double& y){
        double aux = x;
        x = y;
        y = x;
}

int main(){
        double x = 7.0;
        double y = 5.0;
        troca(x, y);
        cout << "X: " << x;
        cout << "Y: " << y;
}
```

- (●) Verdadeiro
- ( ) Falso

A afirmação é verdadeira conforme discutido em sala de aula.

Pergunta 21

0,2 / 0,2 pts

Podemos afirmar que o código abaixo em C++ imprime "1 1" como saída

```
void f (int val, int& ref) {
   val ++; ref++;
```

```
}

void main () {
    int i = 1, j = 1;
    f(i, j);
   printf("%i -- %i", i, j);
}
```

- (●) Falso
- ( ) Verdadeiro

A afirmação é falta. A variável ref foi passada por referência, assim, seu valor final é dois.

Pergunta 22

0,2 / 0,2 pts

Considere a assinatura das duas funções abaixo na linguagem C++:

```
void troca(double &x, double &y)
void troca(double* x, double* y)
```

Podemos afirmar que a primeira função utiliza passagem de parâmetros por referência e a segunda, por valor.

- (●) Verdadeiro
- ( ) Falso

A afirmação é verdadeira considerando os conceitos sobre passagem de parâmetros na linguagem C++ apresentados na sala de aula.

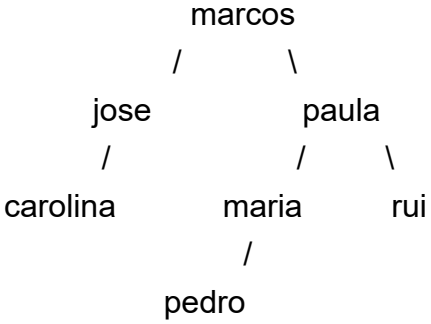
Pergunta 23

0,2 / 0,2 pts

Considere que a Manausprev armazena os nomes dos beneficiários de aposentadorias em uma Árvore Binária de Busca (ABB). Ao se armazenar, nesta ordem, os nomes Marcos, José, Carolina, Paula, Rui, Pedro e Maria, a ABB resultante é uma árvore completa (MANAUSPREV'15, adaptada).

- Verdadeiro
- Falso

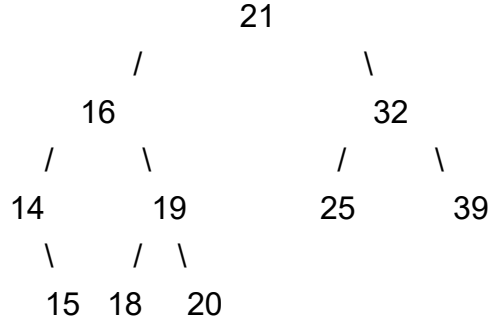
A afirmação é falsa e a árvore resultante é mostrada abaixo.



Pergunta 24

0,2 / 0,2 pts

Dada a árvore binária abaixo e o código do método pesquisar, podemos afirmar que a chamada do método pesquisar(18) imprime na tela duas vezes a letra "A".



```
public boolean pesquisar(int x) {
     return pesquisar(x,raiz);
}

private boolean pesquisar(int x, No i) {
    boolean encontrado;
    if(i == null) {
        encontrado = false;
    } else if(x == i.elemento) {
        encontrado = true;
    } else if(x < i.elemento ) {
        encontrado = pesquisar(x, i.esq);
        System.out.println("A");
    } else {
        encontrado = pesquisar(x, i.dir);
    }
    return encontrado;
}
```

- Falso
- Verdadeiro

A afirmação é verdadeira. A impressão acontece da letra acontece quando o algoritmo caminha para a esquerda a partir dos nós contendo os números 21 e 19.

Pergunta 25

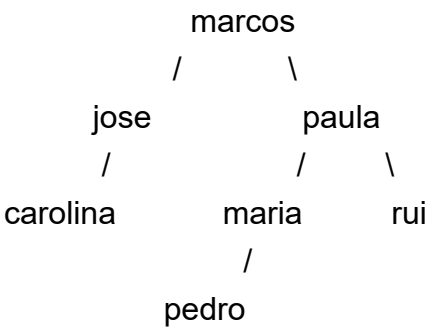
0,2 / 0,2 pts

Considere que a Manausprev armazena os nomes dos beneficiários de aposentadorias em uma Árvore Binária de Busca (ABB). Ao se armazenar, nesta ordem, os nomes Marcos, José, Carolina, Paula, Rui, Pedro e Maria, a ABB resultante tem o nó contendo José possui dois filhos (MANAUSPREV'15, adaptada).

- Verdadeiro

Falso

A afirmação é falsa e a árvore resultante é mostrada abaixo.



Pergunta 26

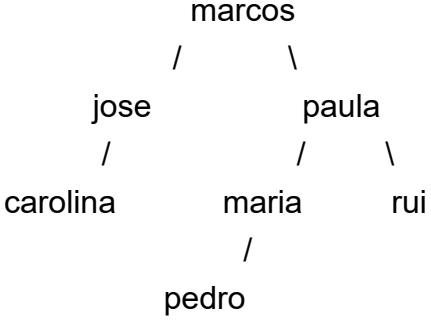
0,2 / 0,2 pts

Considere que a Manausprev armazena os nomes dos beneficiários de aposentadorias em uma Árvore Binária de Busca (ABB). Ao se armazenar, nesta ordem, os nomes Marcos, José, Carolina, Paula, Rui, Pedro e Maria, a ABB resultante tem altura 3, que corresponde à altura mínima para armazenar os 7 nomes (MANAUSPREV'15, adaptada).

Verdadeiro

Falso

A afirmação é falsa e a árvore resultante é mostrada abaixo.



Pergunta 27

0,2 / 0,2 pts

O comando Java “No no = new No()” cria um objeto do tipo No e armazena seu endereço na variável no. A mesma coisa acontece em C++ no código “No no = new No()”.

- (●) Falso
- ( ) Verdadeiro

O comando C++ apresentado não compila, pois "No no" cria um objeto do tipo No e, não, um ponteiro. Logo, a variável no não armazena um endereço de memória como no comando Java apresentado.

Pergunta 28

0,2 / 0,2 pts

Nas linguagens Java e C++, os atributos de uma classe podem ter as visibilidades public, private ou protected. Um atributo público é acessível dentro e fora da classe. Um privado, somente dentro da classe. Um protegido, somente dentro da classe ou de classes filhas.

- (●) Verdadeiro
- ( ) Falso

As linguagens Java e C++ são orientadas por objetos e possuem os três tipos de visibilidade apresentados com suas respectivas definições.

Pergunta 29

0,2 / 0,2 pts

O código C++ abaixo tem um objeto chamado "quadrado_teste" do tipo Quadrado que pode ser usado globalmente.

```
class Quadrado {
   private:
      int lado;
   public:
      void set_values (int);

   //...
} ;

int main() {
   Quadrado quadrado_teste;

   //...
   return 0;
}
```

- (●) Falso
- ( ) Verdadeiro

A declaração da classe e a criação do objeto foram feitos de maneira adequada. Entretanto, como o objeto foi declarado dentro da função main, o escopo de vida do mesmo é aquela função a partir do seu ponto de declaração.

Pergunta 30

0,2 / 0,2 pts

Nas linguagem Java e C++, ao instanciarmos um objeto usando new, fazemos com que tal objeto seja acessível a partir de um ponteiro/referência. Entretanto, diferente do Java, a linguagem C++ também permite a criação literal de objetos.

- (●) Verdadeiro
- ( ) Falso

Nas linguagem Java e C++, o comando new cria um objeto e retorna seu endereço de memória que deve ser armazenado em um ponteiro/referência. A linguagem Java não permite a criação literal de um objeto o que é possível em C++.

Pergunta 31

2,5 / 2,5 pts

Seja a árvore Trie apresentada na sala de aula modifique o método INSERIR supondo que nossa árvore permite a existência de prefixos. Por exemplo, assim, podemos ter as palavras AMO, AMOR, AMORA, AMORES. Apresente a ordem de complexidade do seu método. Sua resposta deve conter somente o método solicitado e sua complexidade.

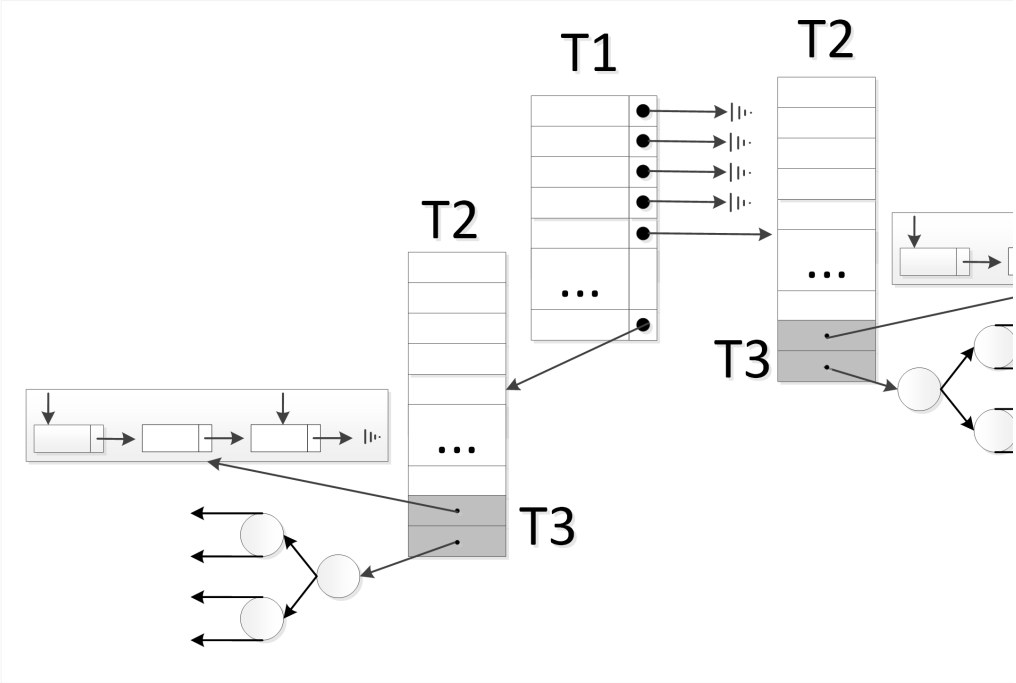
Questao31.java (https://pucminas.instructure.com/files/3895806/download)

Pergunta 32

2,5 / 2,5 pts

Considere a estrutura de dados Doidona ilustrada abaixo. O primeiro nível dessa estrutura contém uma tabela hash T1 sendo que cada célula dessa tabela possui um número inteiro elemento e um ponteiro T2 para outra tabela hash, T2. Cada célula de T1 aponta para uma T2. O tamanho de T1 é tam1 e de cada T2, tam2. Distribuímos os elemento em T1 com uma função int hashT1(int x) que retorna (x % tamT1). A T2 de cada célula distribui seus elementos com uma função int hashT2(int x) que retorna (x % tamT2) e trata as colisões com uma função int rehashT2(int x) que retorna (++x % tam2). Quando a função rehash} não consegue tratar as colisões, a T2 de cada nó trata essas

colisões com a tabela hash} T3 (logicamente implementada). A função int hashT3(int x) retorna (x % 2). Quando esse retorno é zero, inserimos o valor em uma lista listaT3. Quando um, em uma árvore binária arvT3. Implemente o algoritmo para pesquisar a existência de um elemento cadastrado nessa estrutura. Você N.Ã.O deve usar as estruturas de Árvore Binária nem de Lista, use apenas o Nó da Árvore e a Célula da Lista. A classe Doidona tem o atributo CelulaT1 t1[]. A classe CelulaT1 tem os atributos int elemento e HashT2 t2. A classe HashT2 tem os atributos int[] tab; Celula primeiroT3, ultimoT3; e No raizT3. As classes Celula e nó são iguais as vistas na sala de aula. Faça a análise de complexidade do seu método.



Questao32.java
(https://pucminas.instructure.com/files/3895990/download)

ok

Pontuação do teste: **10,8** de 11