|  |  |  |
| --- | --- | --- |
| C:\Users\tamires.mendes\OneDrive - FUNDAÇÃO MOVIMENTO DIREITO E CIDADANIA\Área de Trabalho\2022\_LOGO-DH\Marca_DomHelder_Vertical.png SIMULADO | | |
|  | | |
| **Disciplina: Algoritmos e Estrutura de Dados II** |  |  |
| **Professor: Diego Silva Caldeira Rocha** | | |
| **Aluno(a):** | | |
|  |  |  |
| **Instruções:**   |  | | --- | | 1. Prova individual e sem consulta;  2. Preencha o cabeçalho com seu nome legível e assine nos lugares indicados;  3. As “vistas” (gabaritos) dessa avaliação acontecerão via espelho de correção, postados na aba Conteúdos;  4. Se utilizarem a calculadora do celular, coloque-o no modo avião durante toda a avaliação;  5. A avaliação poderá ser resolvida a lápis, porém dê a resposta final a tinta. As questões com resposta final a lápis não terão direito à revisão;  6. A consulta a colegas ou a qualquer outro material, quando não autorizada pelo professor, será considerada “cola”. Neste caso, o aluno receberá nota zero, cabendo ainda as medidas disciplinares previstas;  7. Utilize apenas o espaço designado para as questões abertas;  8. A resposta de cada questão aberta não poderá ultrapassar o espaço proposto;  9. Durante a aplicação da avaliação não será permitido qualquer tipo de comunicação entre os examinados e nem empréstimo de material;  10. É proibido chamar o examinador para tirar qualquer dúvida, a não ser que haja um erro substancial na avaliação prejudicando o entendimento de uma pergunta;  11. As notas serão lançadas em até 24h, a partir do final dessa avaliação;  12. Apenas serão consideradas as questões respondidas com raciocínio detalhado, de forma clara, organizada e sem rasuras;  13. Se necessário, manifeste seu recurso pelo e-mail  14. Não permaneçam na porta da sala ao finalizarem as avaliações;  15. Não é permitido retirar o grampo das páginas. | | | |

1. Assinale das seguintes afirmações sobre o crescimento assintótico das funções marque **V** para **verdadeira** e **F**  para **falso** (**4pts**):

( V ) .   
( V) .  
( V)

( V)   
( V) .  
( V) .

2. Sejam duas funções f(n) e g(n) que mapeiam números inteiros positivos em números reais positivos. Com respeito às notações assintóticas de complexidade, marque **V** para verdadeiro e **F** para Falso.

**(V)** Diz-se que *f*(*n*) é *O*(*g*(*n*)) se existe uma constante real *c >* 0 e existe uma constante inteira *n*0 *≥* 1 tal que *f*(*n*) *≤ c × g*(*n*) para todo inteiro *n ≥ n*0.

**(F)** Diz-se que *f*(*n*) é *o*(*g*(*n*)) se para toda constante real *c >* 0 existe uma constante inteira  
*n*0 *≥* 1 tal que *f*(*n*) *< c × g*(*n*) para todo inteiro *n ≥ n*0.

**(V)** . Diz-se que *f*(*n*) é Ω(*g*(*n*)) se existe uma constante real *c >* 0 e existe uma constante  
inteira *n*0 *≥* 1 tal que *f*(*n*) *≥ c × g*(*n*) para todo inteiro *n ≥ n*0.

**(F)** . Diz-se que *f*(*n*) é *ω*(*g*(*n*)) se para toda constante real *c >* 0 existe uma constante  
inteira *n*0 *≥* 1 tal que *f*(*n*) *> c × g*(*n*) para todo inteiro *n ≥ n*0.

**(V)**. Diz-se que *f*(*n*) é Θ(*g*(*n*)) se, e somente se, *f*(*n*) é *O*(*g*(*n*)) e *f*(*n*) é Ω(*g*(*n*))

3, Cite os 3 princípios paradigmas de proteção (Princípios de McConnell¹).

1º Garbage in, nothing out (Lixo entra, nada sai)

2º Garbage in, error message out (Lixo entra, mensagem de erro sai)

3º No garbage allowed in (Nenhum lixo é permitido)

4. Quais são os 4 passos que devem ser compreendidos para lançar uma exceção, descreva suscintamente cada um deles.

1º A classe onde e exceção será criada deve estender da classe Exception;

Ex: public class Exceção extends Exception{...}

2º No método onde a exceção será propagada, na declaração no método é necessário utilizar o “throws”;

Ex: public void calculaNumeroInteiro throws Exceção(){...}

3º Para lançar o método é necessário criar uma nova exceção, utilizando o throw new (nome da classe de exceção);

Ex: throw new Exceção(...)

4º Após isso é necessário tratar a exceção utilizando o try/catch

5. Encontre a ordem de complexidade do custo das atribuições dos seguintes trechos de código (**5 pts**):

(a)

for(k = 0; k < n - 2; k++)  
 for(i = k + 1; i <= n; i++){  
 x = vet[k][i];  
 vet[k][i] = vet[i][k];  
 vet[i][k] = x;  
 }

|  |
| --- |
| RESP: O(n2) |

(b)

i = 0;  
 while(i < n){  
 j = 0;  
 while(j <= n){  
 a[i][j] = b[i][j] + c[i][j];  
 j++;  
 }  
 i += 2;  
}

|  |
| --- |
| RESP: O(n2) |

(c)

for(x = 0, j = 1; j <= n; j++)  
 for(i = 1; i <= j ; i++)  
 ++x;

|  |
| --- |
| RESP: O(n2) |

(d)

public static void funcao(int number){  
 int i, j, k;  
 for (i = 1; i < 4; i++)  
 for (j = i; j < n + 1; j++)  
 for (k = i; k < j + 1; k++)  
 number = number + i + j + k;  
}

|  |
| --- |
| RESP: O(n) |

6. Considere um sistema para registros de empregados de uma empresa. O sistema deverá registrar dados representados pelas seguintes classes (**não é necessárior cria a main**):

- Classe Empregado: id (inteiro), nome (string) e salário (float).

- Subclasse *Gestor* especialização da classe *Empregado* (subclasse ou derivada): nível(inteiro), número do celular corporativo (string)

a) Implemente as classes com os atributos e construtores, com os parâmetros relativos aos atributos das classes.

b) Crie métodos *get* e *set* para os atributos da classe Empregado e Gestor.

c) Lance uma exceção (classe Exception) se caso o valor atribuído ao salário for menor ou igual a zero.

|  |
| --- |
| Arquivo Anexado |

7. Crie um método em JAVA que esteja preparado para capitular a exceção do exercício anterior.

|  |
| --- |
| Arquivo Anexado |