SERVIÇO PÚBLICO FEDERAL
MEC / SETEC

INTITUTO FEDERAL
DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
Sul-rio-grandense

Departamento de Seleção

CIDADES DE CAMAQUÃ, CHARQUEADAS, GRAVATAÍ, NOVO HAMBURGO E SAPIRANGA

INSTRUÇÕES GERAIS

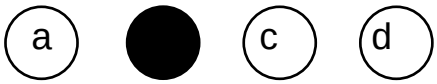
1 - Este caderno de prova é constituído por 40 (quarenta) questões objetivas.

2 - A prova terá duração máxima de 04 (quatro) horas.

3 - Para cada questão, são apresentadas 04 (quatro) alternativas (a – b – c – d). **APENAS UMA delas** responde de maneira correta ao enunciado.

4 - Após conferir os dados, contidos no campo Identificação do Candidato no Cartão de Resposta, assine no espaço indicado.

5 - Marque, com caneta esferográfica azul ou preta de ponta grossa, conforme exemplo abaixo, no Cartão de Resposta – único documento válido para correção eletrônica.



6 - Em hipótese alguma, haverá substituição do Cartão de Resposta.

7 - Não deixe nenhuma questão sem resposta.

8 - O preenchimento do Cartão de Resposta deverá ser feito dentro do tempo previsto para esta prova, ou seja, 04 (quatro) horas.

9 - Serão anuladas as questões que tiverem mais de uma alternativa marcada, emendas e/ou rasuras.

10 - O candidato só poderá retirar-se da sala de prova após transcorrida 01 (uma) hora do seu início.

BOA PROVA!

CONHECIMENTOS ESPECÍFICOS

1. Considere o trecho de código em JavaScript abaixo. Ele utiliza o operador spread (...), que é amplamente usado para manipular arrays e objetos, permitindo a expansão de seus elementos. Observe como o operador interage com o array fornecido.

```
let [a, ...b] = [1, 2, 3, 4];
console.log(b);
```

Após a execução do código, qual será o conteúdo da variável b?

a) [1, 2, 3]
b) [2, 3, 4]
c) 2
d) [1, 2, 3, 4]

2. Considere o seguinte código JavaScript, que cria dinamicamente uma modal no DOM ao clicar em qualquer botão com a classe .openModal. A modal contém um botão de "Fechar", que a remove do DOM quando clicado.

```
document.querySelectorAll('.openModal')
  .forEach(btn => btn.addEventListener('click', () => {
    const modal = document.createElement('div');
    modal.classList.add('modal');
    modal.innerHTML = `<div class="modal-content">
      <h2>Hello World!</h2>
      <p>This is my website</p>
      <button id="closeModal">Fechar</button>
    </div>`;
    modal.querySelector('#closeModal')
      .addEventListener('click', () => modal.remove());
    document.body.appendChild(modal);
}));
```

Com base no código acima, analise as seguintes afirmações sobre o comportamento esperado do site e identifique a **INCORRETA**:

a) Sempre que um botão com a classe .openModal for clicado, uma nova modal será criada e adicionada ao DOM, e o botão "Fechar" funcionará corretamente, removendo a respectiva modal.
b) Mesmo após a primeira modal ser fechada, a próxima modal aberta funcionará corretamente, pois o evento de "Fechar" é vinculado a cada nova instância do botão #closeModal dentro da modal.
c) O botão "Fechar" da modal remove-a do DOM sem deixar vestígios de eventos ou elementos relacionados à modal removida, mantendo o DOM limpo após o fechamento.
d) Se o usuário clicar múltiplas vezes em um botão com a classe .openModal, apenas a última instância da modal permanecerá no DOM.

3. Considere o código a seguir, que manipula um array utilizando a linguagem JavaScript. Sabendo que os arrays em JavaScript não possuem alocação pré-definida e não exigem a declaração prévia de seu tamanho, analise o comportamento do código.

```
const numbers = [1, 2, 3];
numbers[5] = 6;
console.log(numbers.length);
```

Após a execução do código, qual será o valor exibido no console?

a) 3
b) 4
c) 5
d) 6

4. As funções em JavaScript permitem a passagem de valores durante sua execução, possibilitando a manipulação dinâmica de dados e comportamentos com base nos argumentos fornecidos. Considere o seguinte código em JavaScript, que faz a chamada de uma função com parâmetros:

```
function func1(msg, num) {
  return msg+num;
}
let msg = "10";
let num = 10;
let result = func1(msg,num);
console.log(result);
```

Qual será o resultado apresentado após a execução do código?

a) O código apresentará um erro.
b) '1010'
c) 20
d) '20'

5. Observe o trecho de código a seguir, que faz uso dos comandos async/await na linguagem JavaScript. Esses comandos são utilizados para trabalhar com operações assíncronas de maneira mais simples e legível.

```
async function getTodoData() {
  try {
    const response = await fetch('https://jsonplaceholder.typicode.com/todos/1');
    const data = await response.json();
    console.log(data.title);
  } catch (e) {
    console.log("Erro ao buscar dados");
  }
}
getTodoData();
console.log("Depois de getTodoData");
```

O código faz uma requisição à URL 'https://jsonplaceholder.typicode.com/todos/1' e, caso a resposta seja recebida com sucesso, o conteúdo será

```
{"userId":1,"id":1,"title":"delectus aut autem","completed":false}
```

Quais serão as duas saídas apresentadas no console quando o código for executado?

a) undefined, "Depois de getTodoData"
b) "Depois de getTodoData", "delectus aut autem"
c) "delectus aut autem", "Depois de getTodoData"
d) "Erro ao buscar dados", "Depois de getTodoData"

6. No JavaScript, os comandos *break* e *continue* são utilizados para controlar o fluxo de execução em estruturas de repetição, como *for, while* e *do...while*, bem como em estruturas de seleção *switch*.

Qual das afirmações que seguem descreve corretamente o comportamento desses comandos?

a) O comando *break* interrompe a execução do laço e sai imediatamente da estrutura de repetição, enquanto o comando *continue* também encerra a execução do laço, mas apenas após terminar a iteração atual.
b) O comando *break* e o comando *continue* têm o mesmo comportamento, ambos encerram a execução do laço, saindo imediatamente da estrutura de repetição.
c) O comando *break* interrompe o laço completamente e sai da estrutura de repetição, enquanto o comando *continue* faz com que o laço pule a iteração atual e prossiga com a próxima.
d) O comando *break* é utilizado apenas em estruturas *switch*, enquanto o *continue* é utilizado somente em laços *for*, *while* e *do...while*.

7. Analise o seguinte código em JavaScript, que manipula arrays utilizando os métodos sort, map e filter. Esses métodos são amplamente utilizados para ordenar, transformar e filtrar elementos de arrays de forma eficiente, proporcionando uma maneira funcional e concisa de trabalhar com coleções de dados em JavaScript.

```
const numeros = [10, 5, 8, 1, 7];
const final = numeros
  .sort((a,b) => b - a)
  .map(num => num * 3)
  .filter(num => num % 2 === 0);
console.log(final);
```

O conteúdo do array final após a execução do código será:

a) [30, 24, 21, 15, 3]
b) [3, 15, 21, 24, 30]
c) [30, 24]
d) [24, 30]

8. Analise o código em JavaScript, que faz uso da palavra reservada this no contexto de um objeto, em funções regulares e funções arrow. A palavra this refere-se ao contexto de execução em que uma função é chamada, mas seu comportamento pode variar entre funções regulares e funções arrow.

```
const pessoa = {
  nome: 'Alice',
  saudar: function() {
  const saudacaoRegular = function() {
        console.log(`Olá, meu nome é ${this.nome}`);
  };
  const saudacaoArrow = () => {
        console.log(`Olá, meu nome é ${this.nome}`);
  };
  saudacaoRegular();
  saudacaoArrow();
  }
};
pessoa.saudar();
```

A saída no console da execução do código apresentado será:

a) Olá, meu nome é undefined
   Olá, meu nome é Alice
b) Olá, meu nome é undefined
   Olá, meu nome é undefined
c) Olá, meu nome é Alice
   Olá, meu nome é Alice
d) Olá, meu nome é Alice
   Olá, meu nome é undefined

9. Os laços de repetição em JavaScript são usados para iterar sobre coleções de dados, como *arrays*, objetos ou *strings*, permitindo a execução repetida de um bloco de código. Dois dos principais laços usados para esse propósito são *for...in* e *for...of*. Cada um deles possui uma funcionalidade específica e um comportamento distinto na iteração de elementos.

Com relação ao tema apresentado, analise as afirmativas a seguir:

I. O laço *for...of* é usado para iterar diretamente sobre os valores de objetos iteráveis, como *arrays e strings*.
II. O laço *for...in* é usado para iterar sobre as propriedades enumeráveis (chaves) de um objeto.
III. O laço *for...of* pode ser utilizado para iterar sobre as propriedades de objetos que não sejam iteráveis.
IV. O laço *for...in* retorna os valores dos elementos de um *array*, enquanto *for...of* retorna os índices do *array*.

Estão corretas apenas as afirmativas

a) I e II.
b) I e III.
c) II e III.
d) II, III e IV.

10. Em JavaScript, os métodos filter e map são frequentemente usados para manipulação de arrays, enquanto o operador spread (...) pode ser utilizado para copiar, combinar ou transformar arrays e objetos.

Considere o seguinte código que faz uso desses conceitos.

```
const products = [
  { id: 1, name: 'Laptop', price: 1500, available: true },
  { id: 2, name: 'Phone', price: 800, available: false },
  { id: 3, name: 'Tablet', price: 600, available: true }
];
const availableProducts = products
  .filter(product => product.available)
  .map(product => ({ ...product, price: product.price * 0.9 }));
const finalProductList = [
  ...availableProducts,
  { id: 4, name: 'Monitor', price: 300, available: true }
];
console.log(finalProductList[1]);
```

Considerando o código acima, qual das alternativas a seguir descreve corretamente o que será impresso no console após a execução do código?

a) {id: 3, name: 'Tablet', price: 540, available: true}
b) {id: 2, name: 'Phone', price: 720, available: true}
c) {id: 4, name: 'Monitor', price: 300, available: true}
d) {id: 4, name: 'Monitor', price: 270, available: true}

11. Os comandos SQL INSERT, DELETE e UPDATE são utilizados para manipular dados em um banco de dados.

Com relação ao uso desses comandos, é **INCORRETO** afirmar que o comando

a) INSERT permite adicionar registros a uma tabela, independentemente das restrições de integridade impostas pelo banco de dados.
b) INSERT pode ser usado para adicionar múltiplos registros a uma tabela em um único comando.
c) DELETE, quando utilizado sem a cláusula WHERE, remove todos os registros de uma tabela, mas a definição da tabela permanece no banco de dados.
d) UPDATE permite modificar os valores dos atributos de vários registros em uma tabela, e a cláusula SET define os novos valores dos atributos.

12. No contexto de bancos de dados, uma *view* (ou visão) é uma tabela virtual que exibe dados resultantes de uma consulta SQL. Ela não armazena dados permanentemente, exceto quando se trata de *views* materializadas, que guardam fisicamente o resultado da consulta. As *views* são amplamente usadas para facilitar o acesso a informações e restringir a visualização de dados sensíveis.

Com base nisso, analise as afirmações a seguir, assinalando V, para as Verdadeiras, e F, para as Falsas:

( ) Uma *view* é uma tabela virtual que gera seus dados dinamicamente no momento da consulta, sem armazená-los fisicamente.

( ) As *views* podem ser utilizadas para limitar o acesso às colunas ou linhas sensíveis de uma tabela, sem a necessidade de modificar a tabela original.

( ) Ao contrário das tabelas, as *views* permitem atualizações diretas em seus dados, sem nenhuma restrição.

( ) *Views* materializadas armazenam fisicamente os resultados da consulta e precisam ser atualizadas manual ou automaticamente quando os dados das tabelas subjacentes são alterados.

A sequência correta, de cima para baixo, é:

a) V – F – V – F.
b) V – V – F – V.
c) F – V – V – F.
d) F – F – V – V.

13. Um profissional da área de tecnologia da informação está projetando um sistema de gerenciamento escolar para um Instituto Federal de Educação, Ciência e Tecnologia. O sistema deve automatizar várias tarefas e garantir a integridade dos dados por meio do uso de *triggers* no banco de dados. Cada tipo de *trigger* possui um papel específico na execução de operações, como inserções, atualizações e exclusões.

Faça a associação correta para cada tipo de *trigger* na coluna I com a descrição e comportamento correspondente na coluna II.

Coluna I - Tipos de *Trigger*:

1. *Trigger* BEFORE
2. *Trigger* AFTER
3. *Trigger* INSTEAD OF
4. *Trigger* FOR EACH STATEMENT

A. Para garantir que as notas inseridas para os alunos sejam validadas antes de serem gravadas no sistema, essa *trigger* executa ações antes da operação de inserção ou atualização, permitindo verificar ou ajustar os dados.

B. Após a inserção de um novo registro de aluno, é necessário atualizar automaticamente a lista de turmas e registrar o novo aluno em outras tabelas relacionadas. Essa *trigger* executa ações automaticamente depois que a operação principal é concluída.

C. Para enviar um relatório para a coordenação sempre que um comando SQL afeta várias linhas, como a atualização de notas para todos os alunos de uma turma, é utilizada uma *trigger* que realiza a ação para toda a operação em vez de para cada linha individualmente.

D. Se o objetivo é substituir a operação padrão de atualização das notas por uma nova lógica que ajusta a média final dos alunos, essa *trigger* permite definir uma ação alternativa que será executada no lugar da operação original.

Qual é a associação correta entre números e letras?

a) 1-B, 2-C, 3-A, 4-D.
b) 1-C, 2-D, 3-B, 4-A.
c) 1-D, 2-A, 3-C, 4-B.
d) 1-A, 2-B, 3-D, 4-C.

14. No contexto de consultas em banco de dados relacionais utilizando a linguagem SQL, uma consulta pode ser estruturada com diversas cláusulas, sendo que apenas as cláusulas SELECT e FROM são obrigatórias para que a consulta seja válida.

Considerando as cláusulas opcionais e o processo de otimização de consultas, analise as afirmativas a seguir:

I. A cláusula WHERE é usada para definir as condições de seleção de tuplas, incluindo condições de junções se necessário.
II. A cláusula GROUP BY pode ser utilizada sem a presença de funções agregadas, como COUNT e SUM.
III. A cláusula HAVING é aplicada após a cláusula GROUP BY e define condições para selecionar grupos de resultados.
IV. A consulta SQL pode ser otimizada com o uso mínimo de aninhamentos e ordenações.
V. O Sistema de Gerenciamento de Banco de Dados (SGBD) sempre processa consultas da mesma forma, independentemente de como foram formuladas.

Estão corretas apenas as afirmativas

a) II, III e V.
b) I, III e IV.
c) I, III, IV e V.
d) II e V.

15. Uma coleção de operações no banco de dados geralmente é vista como uma única ação pelo usuário. Por exemplo, uma transferência entre uma conta corrente e uma conta poupança parece uma única operação, mas, no banco de dados, envolve várias etapas. Essas coleções de operações são chamadas de transações, e o sistema de banco de dados deve garantir sua execução correta, mesmo em situações de falha.

Com base nos conceitos de transações e recuperação de falhas da linguagem SQL, analise as afirmativas a seguir e, assinale V, para as Verdadeiras, e F, para as Falsas:

( ) Em sistemas SQL que seguem o protocolo ACID, o sistema deve garantir que, em caso de falha durante uma transação, as alterações parciais realizadas até o momento da falha sejam preservadas para facilitar a recuperação.
( ) Quando uma transação tiver sido confirmada (committed), não é possível desfazer seus efeitos abortando-a. Para desfazer os efeitos de uma transação confirmada, é preciso executar uma transação de compensação.
( ) O mecanismo de ponto de verificação (checkpoint) em um banco de dados SQL permite uma recuperação mais rápida após uma falha, pois os dados até o último checkpoint são considerados estáveis e não precisam ser restaurados a partir do log de transações.
( ) A técnica de rollback é utilizada para desfazer alterações realizadas por uma transação que falhou, retornando o banco de dados ao estado estável do último ponto de verificação.

A sequência correta, de cima para baixo, é:

a) V – F – F – V.
b) F – V – V – F.
c) V – F – V – F.
d) F – V – F – V.

16. A IBM desenvolveu a versão original da linguagem SQL, chamada inicialmente de *Sequel*, na década de 1970. Desde então, a SQL evoluiu e se tornou o padrão para bancos de dados relacionais. A linguagem SQL é composta por diferentes subconjuntos, cada um com comandos específicos para realizar várias operações.

Com base nesses subconjuntos, quais são as partes da linguagem SQL responsáveis, respectivamente, por modificar tuplas em uma relação e por criar esquemas de banco de dados?

a) DQL e DML.
b) DDL e DML.
c) DML e DDL.
d) DML e DQL.

17. A linguagem SQL oferece várias funções para manipulação de *strings* de caracteres, como concatenação, conversão para maiúsculas e minúsculas, extração de *substrings*, entre outras. A combinação de padrões pode ser realizada em *strings*, utilizando o operador `LIKE`, o qual permite buscas flexíveis com base em padrões específicos.

Considerando o uso do operador `LIKE`, a ausência de caracteres de espaço nos padrões apresentados e as operações de igualdade entre *strings*, em que há diferenciação entre maiúsculas e minúsculas, analise as afirmativas a seguir:

I. O padrão 'Intro%' combina com qualquer string começando com "Intro", como 'Introdução' e 'Introdução a Banco de Dados'.
II. O padrão '%Comp%' combina com qualquer string contendo "Comp" como substring, por exemplo, 'Introdução a Computação' e 'Computação Sustentável'.
III. O padrão '_ a _' combina com qualquer string de exatamente três caracteres em que o segundo caractere seja "a".
IV. O padrão '%_ _ _%' combina com qualquer string de pelo menos três caracteres, como 'Banco de Dados', 'IFSul' e 'SQL'.

Estão corretas as afirmativas

a) I, II e III, apenas.
b) I, II e IV, apenas.
c) III e IV, apenas.
d) I, II, III e IV.

18. Considere que há uma tabela chamada `docentes` no banco de dados, a qual armazena informações sobre os professores, incluindo matrícula, nome, departamento e salário. Os registros da tabela são os seguintes:

matricula	nome	departamento	salario
13048	John Smith	Artes	10500
11010	Sarah Johnson	Informática	14500
33314	Michael Brown	Geografia	10000
22022	Emily Davis	Artes	18500
18181	David Wilson	Informática	12000
19888	Brian Hall	Artes	11500
44644	Kevin Harris	Informática	14500
12131	Laura Walker	Informática	15500

Para fazer uma consulta que retorne o nome dos departamentos com mais de um docente, a quantidade de docentes que possuem e sua média salarial, ordenados de forma decrescente pela média salarial, utiliza-se o seguinte script SQL:

```
1  SELECT departamento,
2      _____(*) AS quantidade_docentes,
3      _____(salario) AS salario_medio
4  FROM docentes
5  GROUP BY departamento
6  _____ quantidade_docentes > 1
7  ORDER BY salario_medio DESC;
```

Em sequência, as palavras que completam corretamente as lacunas das linhas 2, 3 e 6, para que o script no padrão SQL seja executado corretamente, são:

a) COUNT – AVG – HAVING
b) COUNT – MEAN – WHERE
c) SUM – MEAN – HAVING
d) SUM – AVG – WHERE

Considere as tabelas `discentes` e `matriculas` para responder às questões 19 e 20.

discentes

id	nome	departamento	creditos
1001	Alice Johnson	Computação	20
1002	Brian Smith	Computação	36
1003	Catherine Lee	História	62
1004	David Wilson	Física	90
1005	Emily Davis	Computação	120
1006	Frank Harris	Artes	14
1007	Grace Clark	Artes	38
1008	Henry Scott	Física	80
1009	Irene Moore	Biologia	68
1010	John Taylor	Biologia	22

matriculas

id	disciplina	semestre	ano	conceito
1001	CS-34	1	2018	A+
1001	CS-22	1	2018	B
1006	ART-10	2	2019	B+
1006	ART-14	2	2019	A-
1006	ART-08	1	2020	A
1003	HIS-40	1	2022	A
1007	ART-14	2	2022	A-
1007	ART-12	2	2022	B+
1009	BIO-02	1	2023	B
1009	BIO-04	2	2023	C+
1009	BIO-28	2	2023	B-
1010	BIO-02	2	2023	B+
1010	BIO-04	2	2023	A-
1002	CS-10	1	2024	F
1005	CS-10	1	2024	B+
1005	CS-18	1	2024	*Null*

19. Considerando as tabelas `discentes` e `matriculas` e as diversas operações admitidas em consultas no padrão SQL, analise as afirmativas a seguir:

I. O script SELECT DISTINCT nome FROM discentes, matriculas WHERE discentes.ID = matriculas.ID AND ano BETWEEN 2020 AND 2023; retornará exatamente 2 registros: Catherine Lee e Grace Clark.
II. O script SELECT DISTINCT nome FROM discentes WHERE nome LIKE "%_a%"; retornará exatamente 2 registros: Catherine Lee e David Wilson.
III. O script SELECT departamento, COUNT(DISTINCT id) AS total FROM discentes NATURAL JOIN matriculas GROUP BY departamento; retornará 4 departamentos com seu respectivo número de discentes distintos matriculados em, pelo menos, uma disciplina.
IV. O script UPDATE discentes SET creditos = creditos + 10 WHERE creditos < (SELECT AVG(creditos) FROM discentes); atualizará o número de créditos de, exatamente, 5 discentes.

Estão corretas as afirmativas

a) I, II e III, apenas.
b) I, II e IV, apenas.
c) III e IV, apenas.
d) I, II, III e IV.

20. A linguagem SQL possui uma série de operações de "junção" (`JOIN`) que permite que o programador escreva algumas consultas de forma mais natural e expresse outras consultas que são difíceis de fazer apenas com o produto cartesiano.

Considerando as tabelas discentes e matriculas e o uso de JOINS em SQL, analise as afirmativas a seguir, assinalando V, para as Verdadeiras, e F, para as Falsas:

( ) O script SELECT nome, disciplina FROM discentes NATURAL JOIN matriculas; retornará exatamente o mesmo que SELECT nome, disciplina FROM discentes, matriculas WHERE discentes.ID = matriculas.ID; pois são equivalentes.

( ) O script SELECT nome, disciplina FROM discentes NATURAL JOIN matriculas; retornará exatamente 15 tuplas.

( ) O script SELECT * FROM discentes NATURAL LEFT OUTER JOIN matriculas; retornará 17 tuplas, incluindo os dados dos discentes 1004 e 1008, que não possuem registros na tabela matriculas.

( ) O script SELECT * FROM matriculas NATURAL LEFT OUTER JOIN discentes; retornará 17 tuplas, incluindo os dados dos discentes 1004 e 1008, que não possuem registros na tabela matriculas.

A sequência correta, de cima para baixo, é:

a) V – V – V – F.
b) V – V – F – V.
c) F – F – V – F.
d) F – F – F – V.

21. Em um sistema desenvolvido para o setor de recursos humanos de uma empresa, as informações de nome e salário de cada colaborador são armazenadas em uma classe chamada "Colaborador". Essas informações não podem ser acessadas diretamente por outras partes do programa, sendo necessário o uso de métodos específicos para alterar ou obter esses dados.

A restrição apresentada na classe "Colaborador" é um exemplo de qual conceito de programação orientada a objetos?

a) Herança, pois permite o compartilhamento de atributos entre diferentes classes.
b) Encapsulamento, pois restringe o acesso direto aos dados e controla as alterações por meio de métodos específicos.
c) Polimorfismo, pois os métodos são adaptados de acordo com o tipo de dados fornecido.
d) Abstração, pois oculta os detalhes da implementação interna da classe.

22. Um programador está desenvolvendo um sistema utilizando o paradigma de programação orientada a objetos para um posto de combustível. Neste contexto, considere a definição de classe a seguir utilizando Linguagem Java.

```
class BombaDeCombustivel{
  int id;
  private float qtde;
  private float precoTotal;
  public static precoPorLitro = 6.10;

  public void setQtde(float qtde){
    this.qtde = qtde;
    this.precoTotal = qtde*this.precoPorLitro;
  }
}
```

Sobre o(os) atributo(s) da classe BombaDeCombustivel, é correto afirmar que

a) id e precoPorLitro têm visibilidade pública, sendo que precoPorLitro é um atributo de classe.
b) id, qtde e precoTotal são visíveis em classes que herdem de BombaDeCombustivel.
c) precoPorLitro é uma constante, e seu valor não pode ser alterado em tempo de execução.
d) precoPorLitro é um atributo de classe, portanto seu valor é compartilhado por todas as instâncias desta classe.

23. Quando um objeto é criado em programação orientada a objetos, o método responsável por inicializar o objeto é conhecido como _____. Esse método deve ter o mesmo nome da _____ e pode ser usado para definir valores iniciais para os atributos do objeto. Por exemplo, em um restaurante que vende comida por quilo, o peso do prato pode ser descontado automaticamente da pesagem total por meio desse método. Caso não haja valores fornecidos, os atributos do objeto assumem _____ que foram definidos previamente.

As palavras que preenchem a sentença, completando-a corretamente, são:

a) construtor – classe – valores padrão.
b) inicializador – classe – dados fixos.
c) inicializador – variável – valores padrão.
d) construtor – função – valores predefinidos.

24. Os padrões de projeto são classificados por dois critérios: finalidade e escopo. O primeiro critério, chamado finalidade, reflete o que um padrão faz. Os padrões podem ter finalidade de criação, estrutural ou comportamental.

Os padrões de criação abstraem o processo de instanciação. Eles ajudam a tornar um sistema independentemente de como seus objetos são criados, compostos e representados.

Um padrão de criação de classe usa a herança para variar a classe que é instanciada, enquanto que um padrão de criação de objeto delegará a instanciação para outro objeto. Os padrões de criação se tornam importantes à medida que os sistemas evoluem no sentido de depender mais da composição de objetos do que da herança de classes.

Algumas vezes, os padrões de criação competem entre si. Por exemplo, há casos em que tanto Prototype (121) como Abstract Factory (95) podem ser usados proveitosamente.

Em outras ocasiões, eles são complementares: Builder (104) pode usar um dos outros padrões para implementar quais componentes são construídos. Prototype (121) pode usar Singleton (130) na sua implementação.

Dessa forma, é importante identificar as características de cada padrão.

Faça a associação correta entre as colunas, relacionando o tipo de padrão e suas características.

1-Abstract Factory (95).	A- Fornece uma interface para criação de famílias de objetos relacionados ou dependentes, sem especificar suas classes concretas.
2-Builder (104).	B- Garante que uma classe tenha somente uma instância e fornece um ponto global de acesso para ela.
3-Factory Method (112).	C- Especifica os tipos de objetos a serem criados, usando uma instância prototípica, e cria novos objetos copiando este protótipo.
4-Prototype (121).	D- Separa a construção de um objeto complexo da sua representação, de modo que o mesmo processo de construção possa criar diferentes representações.
5-Singleton (130).	E- Define uma interface para criar um objeto, mas deixa as subclasses decidirem qual classe a ser instanciada. Permite a uma classe postergar (defer) a instanciação às subclasses.

Qual é a associação correta entre números e letras?

a) 1-A, 2-E, 3-B, 4-C, 5-D.
b) 1-B, 2-E, 3-A, 4-D, 5-C.
c) 1-D, 2-B, 3-E, 4-A, 5-C.
d) 1-A, 2-D, 3-E, 4-C, 5-B.

25. A Programação Orientada a Objetos (POO) é um paradigma de programação que revolucionou a forma como desenvolvemos software. Baseada em conceitos como encapsulamento, herança e polimorfismo, a POO oferece uma abordagem estruturada e modular para o desenvolvimento de sistemas complexos. No campo da educação, especialmente no ensino de POO, pesquisadores têm buscado métodos que facilitem o processo de aprendizagem, já que muitas vezes, alunos enfrentam dificuldades para compreender esses conceitos abstratos.

Texto publicado no XXXVI Congresso da Sociedade Brasileira de Computação diz que

"A Programação Orientada a Objetos (POO) mostra-se um paradigma de programação, influente, pelo qual a maior parte dos cursos da área de computação incluem a POO como parte de seus currículos. No entanto, o ensino de POO não é uma tarefa trivial.

Há alertas sobre as dificuldades encontradas por alunos e professores no ensino de POO; os autores apontam que é difícil para os alunos entenderem conceitos abstratos como classes, instâncias, encapsulamento, herança e construtores.

Apesar de muitos esforços terem sido feitos desde então, as disciplinas de programação ainda são responsáveis por um alto índice de reprovação nos cursos de computação."

Considerando os conceitos de Programação Orientada a Objetos, avalie as seguintes asserções e a relação proposta entre elas:

I. A herança de classes é uma das principais características da Programação Orientada a Objetos (POO). Por meio dessa característica do paradigma POO, um objeto recebe características e comportamentos de outro objeto. Quando estiver criando classes, você vai perceber que essa possibilidade permite o reaproveitamento de código e torna o trabalho mais racional e otimizado.

PORQUE

II. Novas classes podem ser definidas em termos das classes existentes, usando-se herança de classe. Quando uma subclasse herda de uma classe-mãe, ela inclui as definições de todos os dados e operações que a classe-mãe define. Os objetos que são instâncias das subclasses conterão todos os dados definidos pela subclasse e suas classes-mãe, e eles serão capazes de executar todas as operações definidas por esta subclasse e seus "ancestrais".

A respeito dessas asserções, qual é a opção correta?

a) As asserções I e II são proposições verdadeiras, mas a II não é justificativa da I.
b) As asserções I e II são proposições verdadeiras, e a II é justificativa da I.
c) A asserção I é proposição verdadeira, mas a II é uma proposição falsa.
d) A asserção I é proposição falsa, mas a II é uma proposição verdadeira.

26. Os modificadores de variáveis são palavras-chave capazes de alterar a visibilidade desses elementos, restringindo seu acesso para leitura e alteração. Considere as seguintes afirmativas:

I-	Private	Torna a variável visível apenas para sua classe.
II-	Protected	A variável só é visível para a classe que foi criada e suas herdeiras.
III-	Public	Pode ser acessada de qualquer classe.
IV-	Sem modificadores	A variável pode ser usada por todas as classes do programa.

Estão corretas apenas as afirmativas

a) I, II e IV.
b) I, III e IV.
c) I, II e III.
d) II, III e IV.

27. A ligação dinâmica é um dos elementos da Programação Orientada a Objetos (POO) que confere flexibilidade a essa metodologia de desenvolvimento de sistemas.

Com relação ao tema apresentado, analise as afirmativas a seguir:

I. Na ligação dinâmica, um método será executado sempre da mesma maneira, independentemente do objeto ou da interação.
II. Na ligação dinâmica, a vinculação tardia é realizada com base na referência armazenada na variável da superclasse, selecionando o método correspondente em tempo de execução.
III. A ligação dinâmica permite determinar, em tempo de execução, qual versão de um método será chamada com base no objeto.
IV. A ligação dinâmica permite que classes, que compartilham a mesma interface, forneçam suas próprias implementações de um mesmo método.

Estão corretas apenas as afirmativas

a) I, II e III.
b) II e IV.
c) I e IV.
d) II, III e IV.

28. As classes abstratas e concretas são fundamentais para organizar e estruturar sistemas de forma eficiente e flexível, pois promovem a reutilização de código e simplificam os processos de manutenção. O uso dessas classes permite ao desenvolvedor criar sistemas mais robustos, modulares e aderentes aos princípios da Programação Orientada a Objetos (POO).

Tendo como referência o tema classes concretas, classes abstratas e métodos abstratos, analise as afirmativas abaixo, assinalando V, para as Verdadeiras, e F, para as Falsas.

( ) As classes concretas atuam como modelos que definem um conjunto de características e comportamentos que suas subclasses devem seguir, sem estarem diretamente associadas a objetos.

( ) Superclasses abstratas obrigam suas subclasses a sobrescrever e implementar os métodos declarados como abstratos.

( ) Uma subclasse concreta deve fornecer implementações concretas de todos os métodos abstratos herdados da superclasse; caso contrário, será considerada abstrata também.

A sequência correta, de cima para baixo, é:

a) F – V – V.
b) V – V – F.
c) V – F – F.
d) F – F – V.

29. Na Programação Orientada a Objetos (POO), os modificadores de acesso são palavras-chave que controlam a visibilidade e o comportamento de classes, atributos e métodos. Eles podem restringir o acesso a esses elementos, tanto dentro de uma classe quanto entre subclasses ou pacotes, além de definir se pertencem à classe ou à instância.

Sobre o tema modificadores, associe os termos às descrições, utilizando os códigos a seguir:

I. static.
II. public.
III. private.
IV. protected.

( ) Ao ser definido em um elemento, significa que pode ser acessado apenas dentro da mesma classe.

( ) Ao ser definido em um elemento, significa que pode ser acessado apenas entre classes que se relacionam em uma hierarquia de herança.

( ) Ao ser definido em um elemento, significa que pertence à classe, em vez de pertencer a instâncias individuais dessa classe.

( ) Ao ser definido em um elemento, significa que pode ser acessado por qualquer classe, seja dentro ou fora do pacote da classe.

A associação correta, de cima para baixo, é:

a) II - IV - III - I.
b) III - IV - I - II.
c) I - II - IV - III
d) IV - III - I - II.

30. Polimorfismo é um princípio fundamental da Programação Orientada a Objetos (POO) que promove a extensibilidade dos sistemas, permitindo a inclusão de novas classes e funcionalidades com mínimas alterações no código existente. Ele se manifesta em dois tipos principais: o polimorfismo por sobrecarga (estático) e o polimorfismo por sobrescrita (dinâmico).

A respeito do conceito apresentado, avalie as asserções a seguir e a relação proposta entre elas:

I. O polimorfismo por sobrecarga em Java permite que métodos com o mesmo nome executem diferentes comportamentos com base no número ou no tipo de parâmetros fornecidos.

II. No polimorfismo por sobrecarga, a escolha de qual método será invocado é feita em tempo de execução, permitindo maior flexibilidade no comportamento dinâmico dos objetos.

Após feita a análise, é correto afirmar que

a) as duas asserções são proposições verdadeiras, e a segunda é uma justificativa correta da primeira.
b) as duas asserções são proposições verdadeiras, mas a segunda não é uma justificativa correta da primeira.
c) a primeira asserção é uma proposição verdadeira; a segunda é uma proposição falsa.
d) a primeira asserção é uma proposição falsa, já a segunda é uma proposição verdadeira.

31. Em PHP, as funções include e require são usadas para incluir o conteúdo de um arquivo em outro.

Sobre o comportamento dessas duas funções, é correto afirmar que

a) o comportamento das duas funções é idêntico. Sendo que o require foi introduzido a partir do PHP 3 por questões semânticas, e o include ainda é mantido na linguagem para fins de compatibilidade de código, mas é considerada obsoleta e novos projetos devem dar preferência a aplicação da função require.
b) a função include gera um aviso e continua a execução do script se o arquivo não for encontrado, enquanto a função require gera um erro e interrompe a execução do script se o arquivo não for encontrado.
c) o comportamento da função require é semelhante ao comportamento da função include. No entanto, ao executar a função require, é realizada uma verificação se o arquivo já foi incluído, e, em caso afirmativo, a inclusão não é realizada.
d) a função require é utilizada para a inclusão de outros arquivos PHP, enquanto a função include é utilizada apenas para a inclusão de arquivos que contenham código HTML (template da página).

32. A tabela **Livros** a seguir armazena informações sobre uma coleção de livros e está definida em um banco de dados MySQL:

id	autor	Titulo	ano_publicacao
1	Machado de Assis	Dom Casmurro	1899
2	Clarice Lispector	A Hora da Estrela	1977
3	Graciliano Ramos	Vidas Secas	1938
⋮	⋮	⋮	⋮

Considere que, em um script PHP, uma conexão com o banco de dados é aberta utilizando o seguinte trecho de código:

```
...
$db = new mysqli('localhost', 'user', 'pass', 'Livros');

if (mysqli_connect_errno()) {
  echo '<p>Erro: Não foi possível conectar ao BD.</p>';
  exit;
}
```

Qual opção apresenta o trecho de código capaz de exibir a quantidade de livros cadastrados no banco de dados?

a)

```
$query = "SELECT * FROM Livros";
$stmt = $db->prepare($query);
$stmt->execute();
$stmt->store_result();

$nro_livros = $stmt->num_rows;
echo "<p>Livros cadastrados: " . $nro_livros ."</p>';
```

b)

```
$query = "SELECT * FROM Livros";
$result = $stmt->execute($query);

$nro_livros = count($result);

echo "<p>Livros cadastrados: " . $nro_livros ."</p>';
```

c)

```
$query = "SELECT * FROM Livros";
$result = $stmt->execute($query);

$nro_livros = count($result)/4;

echo "<p>Livros cadastrados: " . $nro_livros ."</p>';
```

d)

```
$query = "SELECT * FROM Livros";
$stmt = $db->prepare($query);
$stmt->execute();

$nro_livros = $stmt->num_rows();
echo "<p>Livros cadastrados: " . $nro_livros ."</p>';
```

33. Em sistemas para a internet, muitas vezes se faz necessário o armazenamento de estado entre requisições. Para tanto, servidores web utilizam principalmente dois recursos: variáveis de sessão e cookies.

Considere que, em um script PHP, é executada a linha de código a seguir:

```
setcookie('xxx', 'yyy', time()+30);
```

Considere as seguintes asserções:

I. A linha cria um cookie com o nome '**xxx'** armazenando nele a string '**yyy'**.
II. O cookie definido pela linha de código em questão irá expirar em 30 minutos após a sua criação.
III. O valor armazenado no cookie pode ser acessado, utilizando a seguinte chamada de função: getcookie('xxx').
IV. O valor armazenado no cookie pode ser acessado através da superglobal $_COOKIE, da seguinte forma: $_COOKIE['yyy'].

Está(ão) correta(s) apenas a(s) seguinte(s) asserção(ões)

a) I e II.
b) II e IV.
c) I e III.
d) I.

34. O formato JSON (JavaScript Object Notation) é uma ferramenta poderosa e amplamente utilizada para manipulação e troca de dados estruturados. A Linguagem PHP oferece suporte nativo ao formato, sem necessidade de instalações adicionais ou configurações especiais.

Considerando o suporte nativo da linguagem, analise o script em PHP que segue.

```
<?php
$data = [
   "nome" => "João",
   "idade" => 30,
   "habilidades" => ["PHP", "JavaScript", "SQL"],
   "ativo" => true,
   "saldo" => null
];

$json = json_encode($data, JSON_PRETTY_PRINT | JSON_UNESCAPED_UNICODE);
echo $json;
?>
```

Qual das opções apresenta a saída do código acima?

a)
```
{
   "nome": "Jo\u00e3o",
   "idade": 30,
   "habilidades": [
      "PHP",
      "JavaScript",
      "SQL"
   ],
   "ativo": true,
   "saldo": null
}
```

b)
```
{
   "nome": "João",
   "idade": 30,
   "habilidades": ["PHP", "JavaScript", "SQL"],
   "ativo": true,
   "saldo": null
}
```

c)
```
{
   "nome": "João",
   "idade": 30,
   "habilidades": [
      "PHP",
      "JavaScript",
      "SQL"
   ],
   "ativo": true,
   "saldo": null
}
```

d)
```
{
   "nome": "Jo\u00e3o",
   "idade": 30,
   "habilidades": ["PHP", "JavaScript", "SQL"],
   "ativo": true,
   "saldo": null
}
```

35. Considere que o script PHP a seguir é executado em um servidor Web com PHP instalado e a extensão GD devidamente configurada. Considere ainda que, no diretório em que está o script em questão, existe um arquivo fonts/font.ttf que armazena um formato de fonte.

```
1  <?php
2
3  function gerarImagem($largura, $altura, $texto) {
4      $imagem = imagecreatetruecolor($largura, $altura);
5      $branco = imagecolorallocate($imagem, 255, 255, 255);
6      $preto = imagecolorallocate($imagem, 0, 0, 0);
7      imagefilledrectangle($imagem, 0, 0, $largura, $altura, $branco);
8      $font_path = 'fonts/font.ttf';
9      $bbox = imagettfbbox(20, 0, $font_path, $texto);
10     $x = ($largura - $bbox[2]);
11     $y = ($altura - $bbox[1]);
12     imagettftext($imagem, 20, 0, $x, $y, $preto, $font_path, $texto);
13     header('Content-Type: image/png');
14     imagepng($imagem);
15     imagedestroy($imagem);
16 }
17
18 gerarImagem(800,600,"Instituto Federal");
```

Com base no código em questão, é correto afirmar que

a) o script renderiza uma imagem PNG com o texto "Instituto Federal" centralizado horizontal e verticalmente.
b) a chamada da função `header` é opcional, pois o interpretador identifica automaticamente o tipo de retorno.
c) a linha 11 poderia ser modificada por `$y = ($altura - $bbox[3]);` e não haveria qualquer mudança no retorno do script.
d) o script renderizará uma imagem PNG em branco, pois a chamada da função `imagedestroy` limpará o canvas.

36. Em uma aplicação Node.js com o framework Express, as rotas HTTP (POST, DELETE, GET, e PUT) são usadas para realizar diferentes operações com recursos. Seguindo as boas práticas de APIs REST, esses verbos representam ações coerentes com o padrão CRUD (Create, Read, Update, Delete), assegurando que a API siga padrões de design que promovem consistência, escalabilidade e legibilidade do código.

Agora, considere o seguinte trecho de código, onde um CRUD básico para produtos está implementado.

```
var express = require('express');
var app = express();

app.get('/products', (req, res) => {
  // implementação da rota
});

app.post('/products', (req, res) => {
  // implementação da rota
});

app.put('/products/:id', (req, res) => {
  // implementação da rota
});

app.delete('/products/:id', (req, res) => {
  // implementação da rota
});

app.listen(3000);
```

Com base no comportamento esperado das rotas HTTP em um CRUD para produtos, qual das afirmativas está **INCORRETA**?

a) Para consultar produtos, deve-se utilizar a rota GET /products. Os parâmetros de filtragem devem ser fornecidos na query string. Exemplo: GET /products?category=electronics.
b) Para remover um produto, deve-se utilizar a rota DELETE /products/:id, onde o id do produto a ser deletado é passado diretamente nos parâmetros da rota.
c) Para alterar um produto existente, deve-se utilizar a rota PUT /products/:id, onde o id do produto é passado nos parâmetros da rota e as novas informações do produto são enviadas no corpo da requisição.
d) Para cadastrar um novo produto, deve-se utilizar a rota POST /products, enviando os dados do produto através da query string.

37. O gerenciamento de pacotes em um projeto Node.js é facilitado pelo npm, que utiliza o arquivo package.json para armazenar informações sobre o projeto, dependências e scripts. O package.json é gerado automaticamente ao iniciar um novo projeto com o comando npm init.

Os scripts definidos na seção "scripts" do package.json permitem automatizar tarefas comuns, como a execução de testes, a construção de projetos e a inicialização do servidor. Esses scripts podem ser executados usando o comando npm run <nome-do-script>. Além disso, o npm permite a instalação de pacotes como dependências de produção ou de desenvolvimento.

Considerando o uso correto do npm e do arquivo package.json, analise as afirmativas a seguir:

I. O comando npm run <nome-do-script> é utilizado para executar scripts personalizados definidos na seção "scripts" do package.json, permitindo a automação de tarefas como testes e construção de projetos. O npm executa esses scripts em um ambiente isolado, onde as dependências do projeto são acessíveis.
II. O arquivo package.json é opcional em projetos pequenos, pois o npm pode gerenciar pacotes instalados globalmente sem a necessidade de um manifesto local.
III. Ao instalar um pacote com o comando npm install --save-dev <pacote>, o pacote será adicionado à seção "devDependencies" do package.json, indicando que é uma dependência de desenvolvimento e não deve ser incluído em builds de produção.
IV. O diretório node_modules, onde os pacotes instalados pelo npm são armazenados, deve ser incluído no controle de versão para garantir que todas as dependências sejam restauradas em ambientes de desenvolvimento e produção.

Estão corretas apenas as afirmativas

a) II e IV.
b) III e IV.
c) I e III.
d) I e II.

38. Em uma aplicação Node.js usando o framework Express, existem três formas principais de capturar dados de uma requisição: req.body, req.params, e req.query. Cada uma dessas formas possui um uso específico. Considere o seguinte código que implementa uma rota de atualização de usuário.

```
var express = require('express');
var app = express();

app.put('/users/:id', (req, res) => {
  const userId = req.params.id;
  const userName = req.body.name;
  const isActive = req.query.active;

  if (!userId || !userName) {
    return res.status(400).send('Missing required parameters.');
  }
  res.send(`
    User ${userId} updated.
    Name: ${userName},
    Active: ${isActive}
  `);
});

app.listen(3000);
```

Com base no código acima, qual das afirmativas abaixo descreve corretamente como os diferentes tipos de parâmetros devem ser usados para obter o comportamento esperado?

a) O parâmetro req.body é usado incorretamente, pois ele não deveria ser utilizado para capturar o nome do usuário em uma requisição PUT. O dado deveria ser enviado como parte da query string.
b) O parâmetro req.params deve ser evitado, pois é menos seguro do que req.body. O ID do usuário deve ser enviado no corpo da requisição para garantir a integridade dos dados.
c) O código está correto, pois req.params captura o ID do usuário da rota, req.body obtém os dados enviados no corpo da requisição (como o nome), e req.query obtém os parâmetros geralmente opcionais, como o status active, enviado na query string.
d) O código deveria enviar todos os parâmetros no corpo da requisição para maior consistência. Usar req.params e req.query introduz complexidade desnecessária e pode dificultar a manutenção.

39. Em uma aplicação Node.js usando o framework Express, é possível criar middlewares para diferentes propósitos, como autenticação, manipulação de requisições, e registro de logs. Suponha que você tenha a seguinte função de middleware de autenticação que verifica se o usuário tem permissão para acessar uma rota específica com base no papel (role) associado ao usuário.

```
function authRoleMiddleware(role) {
  return (req, res, next) => {
    if (req.user.role !== role) {
      return res.status(403).send('Access forbidden: Insufficient permissions');
    }
    next();
  }
}
```

Este middleware é utilizado da seguinte forma para restringir o acesso de rotas a usuários com o papel de "admin":

```
var express = require('express');
var app = express();

app.use('/admin', authRoleMiddleware('admin'));
app.get('/admin', (req, res) => {
  res.send('Welcome, admin!');
});

app.listen(3000);
```

Considerando o funcionamento desse middleware, qual seria um problema que poderia surgir ao aplicá-lo em rotas que utilizam diferentes middlewares, e como esse problema pode ser resolvido?

a) O middleware depende da presença do req.user, que pode não ser definido se não houver outro middleware executado anteriormente que faça a autenticação. Para resolver, deve-se garantir que um middleware de autenticação seja aplicado antes do authRoleMiddleware.
b) O middleware falha em manipular requisições que possuem cabeçalhos complexos. Para resolver, o authRoleMiddleware deve ser reescrito para analisar os cabeçalhos HTTP de forma mais robusta.
c) O middleware pode causar conflitos quando combinado com rotas que possuem múltiplos parâmetros dinâmicos. A solução é usar router.use() em vez de app.get() para evitar que os middlewares se sobreponham.
d) O middleware pode interferir na performance de rotas públicas, mesmo que não exijam autenticação. A solução é aplicar o authRoleMiddleware apenas em rotas com acesso restrito usando o método .all() do Express.

40. Uma aplicação web foi desenvolvida utilizando Node.js com o framework Express. O objetivo é implementar um middleware simples de autenticação para proteger a rota /login. Este middleware verifica o cabeçalho Authorization das requisições HTTP, validando um token pré-definido. Além disso, o código inclui o tratamento de erros de autenticação e uma resposta padrão para rotas inexistentes.

Considere o seguinte trecho de código que implementa essa funcionalidade.

```
var express = require('express');
var app = express();

const auth = (req, res, next) => {
   try {
      const token = req.headers.authorization.split(' ')[1];
      if (token !== 'XXX') {
         throw new Error('ERROR!');
      }
      next();
   }
   catch (err) {
      res.status(401).send({ message: 'Unauthorized' });
   }
}
app.use('/login', auth);
app.post('/login', (req, res) => {
   res.send('Hello World');
});
app.use((req, res) => {
   res.status(404).send({ message: 'Not Found' });
});

app.listen(3000);
```

Com base no comportamento deste código, considere as seguintes requisições HTTP feitas ao servidor:

I. POST /login/auth HTTP/1.1
Authorization: Bearer XXX

II. GET /login HTTP/1.1
Authorization: XXX

III. POST /login HTTP/1.1
Authorization: Bearer XXX

IV. POST /login HTTP/1.1
Authorization: XXX

Dado o código apresentado, qual é o status de resposta esperado para cada uma das requisições listadas acima?

a) 200, 404, 200, 500
b) 404, 200, 401, 200
c) 401, 500, 200, 404
d) 404, 404, 200, 401

FOLHA DE RASCUNHO

21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20