SERVIÇO PÚBLICO FEDERAL
MEC / SETEC

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

DES
Departamento de Seleção

CIDADE DE BAGÉ

INSTRUÇÕES GERAIS

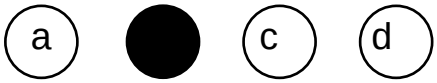
1 - Este caderno de prova é constituído por 40 (quarenta) questões objetivas.

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. Em *JavaScript*, o operador *new* cria e inicializa um novo objeto.

Qual operador **NÃO** representa a criação de um objeto de tipo nativo *JavaScript*?

a) var o = new Object();
b) var l = new ArrayList();
c) var a = new Array();
d) var d = new Date();

2. Considere o seguinte código *JavaScript*:

```
var a = [1,2,3,4,5];
a.slice(0,3);
a.splice(1,1);
a.pop();
```

Qual o valor da variável a ao término da execução do código?

a) [1,3,4]
b) [1,3,4,5]
c) [1,3]
d) [3,4,5]

3. Na versão *draft* da especificação *Web Storage*, são definidas duas propriedades no objeto *Window, localStorage e sessionStorage.*

Sobre o armazenamento de dados em *JavaScript*, usando *localStorage*, afirma-se que

a) dados armazenados por meio do *localStorage* têm a mesma vida útil que a janela de nível superior ou guia do navegador em que o *script* que os armazenou está sendo executado.
b) o escopo de armazenamento do *localStorage* permite que, por exemplo, os dados armazenados em uma visita a um site, usando Firefox, estejam acessíveis quando for realizada uma segunda visita usando o *Chrome*.
c) dados armazenados por meio de *localStorage* têm vida útil diferente de dados armazenados por meio de *sessionStorage.*
d) *localStorage* tem como escopo a origem do documento, que é definida por seu protocolo, porta e criptografia.

4. Considere o seguinte código *JavaScript*:

```
let o = {one:1,two:2,three:3};
for(let p in o) console.log(p);
```

Ao final da execução, quais valores serão impressos?

a) 1, 2, 3
b) one:1, two:2, three:3
c) p, p, p
d) 'one', 'two', 'three'

5. Considere o seguinte código JavaScript:

```
var exemplo = "A";
outro = "A";
function minhaFuncao(){
    var exemplo = "B";
    return exemplo;
}
function minhaFuncao2(){
    outro = "B";
    return outro;
}
console.log(exemplo);
console.log(minhaFuncao());
console.log(outro);
console.log(minhaFuncao2);
console.log(outro);
```

Ao final da execução, quais valores serão impressos?

a) A,B,A,B,A.
b) A,B,A,B,B.
c) A,A,A,B,A.
d) A,A,A,A,A.

6. Considere o seguinte código *JavaScript*, sabendo que o usuário irá digitar corretamente os valores solicitados via *prompt:*

```
var v1 = 3;
var v2,v3,v4;
v2 = prompt("Digite o número 3:");
v2 = prompt("Digite a palavra true:");
v4 = false;

console.log(v1===v2);
console.log(v2==v3);
console.log(v1%=v2);
console.log(v1);
```

Ao final da execução, quais valores serão impressos?

a) false, false, 0, 0
b) false, false, NaN, NaN
c) true, false, NaN, 3
d) true, false, 0, 3

7. A Classe definida no *ECMAScript* 2015, para permitir armazenar valores únicos de qualquer tipo, é

a) Has.
b) ObjectUnique.
c) Set.
d) Unique.

8. A linguagem *JavaScript* provê uma série de métodos que facilitam a manipulação de *arrays.*

Sobre o método de manipulação de *array of*, é correto afirmar que

a) cria um novo *array* a partir de um *array* existente.
b) preenche o *array* com um valor estático.
c) devolve @@*iterator*, contendo os valores do *array*.
d) cria um novo *array* a partir dos argumentos passados para o método.

9. Considerando algoritmos que podem ser usados para percorrer grafos, afirma-se que

a) no algoritmo DFS, ao armazenar os vértices em uma pilha, os vértices serão explorados ao longo de um caminho, visitando um novo vértice adjacente se houver um disponível.
b) no algoritmo BFS, ao armazenar os vértices em uma pilha, os vértices serão explorados ao longo de um caminho, visitando um novo vértice adjacente se houver um disponível.
c) no algoritmo DFS, ao armazenar os vértices em uma fila, os vértices serão explorados ao longo de um caminho, visitando um novo vértice adjacente se houver um disponível.
d) no algoritmo BFS, ao armazenar os vértices em um grafo, os vértices serão explorados ao longo de um caminho, visitando um novo vértice adjacente se houver um disponível.

10. Considere o código de inicialização de variáveis apresentado abaixo: marque a alternativa na qual os valores serão atribuídos, respectivamente, para x e y.

```
let [x,y] = ['a','b','c','d'];
```

Os valores atribuídos para x e y são, respectivamente,

a) 'ac' e 'bd'
b) 'ab' e 'cd'
c) 'a' e 'b'
d) 'c' e 'd'

11. Os *states* do *Phaser* são gerados a partir de classes com métodos específicos para sua execução. Esses métodos são automaticamente chamados pelo *framework* à medida que o jogo é executado.

A sequência de execução de métodos de *state* do *framework Phaser* é

a) preload, load, create, update.
b) init, load, create, update.
c) init, preload, create, update.
d) load, init, create, update.

12. Dentre os sistemas de física em 2D suportados pelo *Phaser*, o mais leve e indicado para jogos mobile é

a) P2.
b) Ninja.
c) Box2D.
d) Arcade.

13. O framework *Phaser* provê uma série de comandos que permitem inserir diversos recursos ao jogo de forma facilitada através da disponibilização de métodos.

No *Phaser*, o comando que insere uma imagem na tela é

a) *game.load.image*('identificacaodaImagem', 'caminhodaimagem')
b) *game.load.image*(posX, posY, 'identificacaodaImagem')
c) *game.add.image*('identificacaodaImagem', 'caminhodaimagem')
d) *game.add.image*(posX, posY, 'identificacaodaImagem')

14. *Node.js* é uma Linguagem baseada no motor de *JavaScript* V8 do *Chrome*.

Quanto a sua orientação e arquitetura, o *Node.js* é uma linguagem que é orientada a

a) objeto e usa um esquema multi-*threading*, bloqueante.
b) objeto e possui um modelo de E/S não bloqueante.
c) eventos e possui uma arquitetura multi-*threading* e bloqueante.
d) eventos e possui um modelo de E/S não bloqueante.

15. O *JavaScript* possui características de uma linguagem funcional, portanto pode-se passar funções como parâmetros para outras funções, algo comumente encontrado nos códigos em *JavaScript* e *Node.*

Sabendo disso, analise as afirmativas abaixo:

I.

```
function soma(a,b){
    return a + b;
}

function executar(funcao,a,b){
    return funcao(a,b)
}

let resultado = executar(soma,1,2)

console.log(resultado)
```

II.

```
function executar(a,b,funcao){
    return funcao(a,b)
}

let resultado = executar(1,2,function(a,b){
    return a+b
})

console.log(resultado)
```

III.

```
var funcao = function(a,b){
    return a+b
}

function executar(a,b,funcao){
    return funcao(a,b)
}

let resultado = executar(1,2,funcao)

console.log(resultado)
```

Está(ão) correta(s) a(s) afirmativa(s)

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

16. Analise as afirmações abaixo sobre declaração de variáveis:

I. *let* declara uma variável presa em um contexto, seja este dentro de uma função, seja dentro de um if.

II. É possível alterar o valor de uma variável *const*.

III. As declarações de variáveis utilizando *var* possuem escopo elevado, conhecido como *hoisting*.

Está(ão) correta(s) apenas a(s) afirmativa(s)

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

17. Sobre o NPM - *Node Package Manager* é correto afirmar que

a) não permite aos desenvolvedores a distribuição de seus pacotes.
b) é necessário o pagamento de uma taxa anual para a disponibilização dos pacotes no npm.
c) é um gerenciador de pacotes global para JavaScript.
d) é o arquivo *package.json* que fica na raiz do projeto e nele são declaradas todas as configurações de banco de dados e senhas de acesso.

18. Analise o código do *package.json* abaixo:

```
{
  "name": "prova",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

Supondo que o desenvolvedor do sistema abra o terminal na pasta desse projeto e execute o comando *npm start,* o que acontecerá?

a) Vai ser lançado o seguinte erro no terminal *npm ERR! missing script: start.*
b) O sistema vai executar o arquivo *main index.js*.
c) O sistema vai tentar instalar as dependências do sistema.
d) Será recriado o arquivo *package.json.*

19. O sistema de módulos do *Node* é modelado a partir do *CommonsJs module system*, uma maneira de criar módulos que garante a interoperabilidade total entre eles. O núcleo desse sistema é um acordo seguido à risca pelos desenvolvedores, para assegurar que seus módulos trabalhem bem com os demais módulos.

Qual **NÃO** contém um requisito do *CommonsJs module system*?

a) Uma função *require,* que recebe o identificador do módulo e devolve a API exportada.
b) O nome do módulo é uma *string* e pode incluir caracteres barra (/) para identificação de caminhos (*paths*).
c) O módulo deve exportar explicitamente o que precisa ser exportado para fora do módulo.
d) As variáveis do módulo devem ser sempre públicas.

20. Sobre a instalação de dependências no *node*, analise as seguintes informações:

I. O parâmetro -g do comando *npm install -g nodemon* é utilizado para instalar a dependência como global. Isso quer dizer que ele vai ser instalado para todos os projetos desenvolvidos neste computador.

II. O parâmetro --*save* é utilizado para salvar a dependência no arquivo *package.json*.

III. A pasta *node_modules* é automaticamente criada pelo npm quando se utiliza o comando *npm init.*

IV. Ao utilizar o comando *npm install* será realizada a instalação de todas as dependências cadastradas no *package.json*.

Estão corretas apenas as afirmativas

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

21. Com relação aos módulos e como eles são carregados na memória, o que é correto afirmar?

a) O Node utiliza a instrução *require* para incluir acesso a um módulo nativo do Node, os módulos não nativos utilizam a instrução *include.*
b) É possível acessar uma propriedade específica de um objeto exportado.
c) Um objeto específico do módulo só pode ser acessado dentro do módulo a que ele pertence.
d) O Node permite mais de um módulo por arquivo.

22. Sobre o Banco de Dados *MongoDB*, analise as afirmações abaixo:

I. *MongoDB* é um banco de dados baseado em documentos, e esses documentos são codificados como BSON – um formato binário do JSON.

II. No *MongoDB,* não existem tabelas, chaves primárias e nem chaves estrangeiras.

III. No *MongoDB*, o equivalente a cada linha de uma tabela do modelo relacional chama-se *Collections*.

Está(ão) correta(s) a(s) afirmativa(s)

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

23. Com relação ao módulo *Express*, é **INCORRETA** a seguinte afirmação:

a) O *Express* é um *framework web* para *Node* que contém um conjunto essencial de funcionalidades para agilizar o desenvolvimento de *web services*.
b) O *Express* possui algumas funções utilitárias no objeto response (res) e uma delas é a res.json([body]), a qual retorna um JSON na resposta.
c) Por padrão, o *Express* não consegue ler as informações enviadas por *POST.* Para isso, deve-se instalar o módulo *body-parser*.
d) Quando uma URL é chamada pelo método GET, pode-se passar parâmetros diretamente na URL. No link http://localhost:3000/pessoa/?nome=Roberto realiza-se a leitura do parâmetro nome usando o objeto *req.params.nome*.

24. O módulo *express* possui um mecanismo chamado *middleware* que é responsável por tratar todas as requisições e mapear as rotas para as funções dentro do código.

Sobre esse mecanismo é correto afirmar que

a) os middlewares de tratamento de erros são utilizados para registrar erros da aplicação, sejam de rotas não encontradas, sejam de exceção de código.
b) os Middlewares são funções que têm acesso ao objeto de requisição (*request*) e ao objeto de resposta (*response*), porém não possuem acesso à próxima função de middleware no ciclo de requisição resposta.
c) o Middleware intercepta a requisição e faz o tratamento desejado. Ele pode terminar a requisição e devolver a resposta, porém ele não pode delegar o trabalho ao próximo middleware.
d) o objeto *next* dos middlewares é utilizado para devolver a resposta da requisição.

25. Os membros de uma superclasse definidos com o modificador *protected* da linguagem Java podem ser acessados por

a) membros dessa superclasse e somente por membros definidos como estáticos na subclasse.
b) membros dessa superclasse e por membros da subclasse.
c) membros dessa superclasse, membros da subclasse e membros de outras classes no mesmo pacote.
d) membros dessa superclasse.

26. Na linguagem Java, a palavra-chave que deve ser usada para especificar que uma variável não pode ser modificada é

a) static.
b) final.
c) const.
d) constant.

27. REST é um estilo de arquitetura de software que define um conjunto de restrições a serem usadas para a criação de web services. **NÃO** é um método de envio válido ao desenvolver *web services* em REST:

a) POST
b) SEND
c) PUT
d) DELETE

28. Existem oito métodos definidos no protocolo HTTP que são: GET, HEAD, POST, PUT, DELETE, TRACE, OPTIONS e CONNECT. Esses métodos indicam a ação a ser realizada no recurso especificado. Analise as afirmações abaixo sobre os métodos GET e POST:

I. GET deve ser usado para obter dados.

II. POST deve ser usado para enviar dados para serem processados.

III. As solicitações GET aceitam que os visitantes façam bookmark da página; as POST não.

IV. Com o POST, o parâmetro é limitado ao que se pode colocar na linha de solicitação.

Estão corretas apenas as afirmativas

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

29. Na hierarquia de exceções em Java, é correto afirmar que

a) a classe *RuntimeException* é uma subclasse da classe *Exception*.
b) a classe *Error herda* da classe *Exception*.
c) as classes NullPointerException e IndexOutOfBoundsException não são válidas no tratamento de exceção em Java.
d) a classe *Exception* é uma subclasse da classe IOException.

30. Independente da linguagem de programação para *web* que o desenvolvedor estiver utilizando, sempre é possível passar parâmetros pela URL.

Qual das formas a seguir é uma forma correta de declaração do método doGet em um servlet Java ?

a) protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException
b) protected void doGet(HttpRequest req, HttpResponse resp) throws ServletException, IOException
c) protected void doGet(HttpServletResponse resp, HttpServletRequest req) throws ServletException, IOException
d) protected void doGet(HttpResponse resp, HttpRequest req) throws ServletException, IOException

31. O *HttpServletResponse* representa a resposta do Servlet. Analise as afirmações abaixo sobre *HttpServletResponse*:

I. setContentType() é um método que diz ao *browser* como tratar os dados enviados na requisição.

II. O método sendResponse(aStringURL) permite redirecionar uma solicitação.

III. Não é possível fazer um redirecionamento após uma resposta ter sido gerada.

Está(ão) correta(s) apenas a(s) afirmativa(s)

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

32. Sobre o arquivo AndroidManifest.xml, é procedente a seguinte afirmação:

a) Dentro da tag <manifest> é declarado o pacote principal do projeto, utilizando a tag <package>.
b) O arquivo AndroidManifest.xml contém a documentação completa da aplicação criada, inclusive, das funções globais da aplicação.
c) O arquivo AndroidManifest.xml é gerado automaticamente ao compilar o projeto e permite que a aplicação acesse qualquer recurso como arquivo e imagens utilizando constantes.
d) No arquivo AndroidManifest.xml são realizadas várias configurações da aplicação e é nele que são inseridos os dados de acesso ao banco de dados.

33. Para iniciar o desenvolvimento de aplicações *Android* é necessário realizar a instalação de alguns softwares e realizar algumas configurações. Sabendo disso, analise as afirmações abaixo sobre o ambiente de desenvolvimento Android:

I. Android SDK é o software utilizado para desenvolver aplicações no Android, que tem um emulador para simular o dispositivo, ferramentas utilitárias e uma API completa para a linguagem Java, com todas as classes necessárias para desenvolver aplicações.

II. Como existem muitas versões do sistema operacional Android, existe um identificador de cada uma dessas plataformas que se chama API Level.

III. *Gradle* é um moderno sistema de gerenciamento de banco de dados para Android.

IV. O Android Studio conta com um utilitário chamado SDK Manager onde é possível baixar todas as plataformas do Android e suas documentações, drive USB do Google para conectar um dispositivo na USB, bibliotecas de compatibilidade, bibliotecas do Google Play Services etc.

Está (ão) correta (s) apenas a (s) afirmativa (s):

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

34. A respeito da classe *Activity,* no desenvolvimento de aplicações Android, analise as afirmações a seguir:

I. A classe Activity deve ser herdada da classe android.app.Activity ou de alguma subclasse desta, a qual representa uma tela da aplicação e é responsável por tratar eventos gerados nessa tela.

II. a classe Activity deve sobrescrever o método onCreate(bundle). Esse método é obrigatório e responsável por realizar a inicialização necessária para executar a aplicação.

III. a classe Activity é uma subclasse da classe FragmentActivity.

IV. a classe AppCompatActivity é uma subclasse da classe Activity.

Estão corretas apenas as afirmativas

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

35. Em relação ao ciclo de vida de classe *Activity*, é procedente afirmar que o método

a) *onStart*() é chamado quando a *activity* está ficando visível ao usuário e já tem uma *view*.
b) *onResume*() é chamado quando uma *activity* foi parada temporariamente e está sendo iniciada outra vez.
c) *onStop*() é chamado sempre que a tela da *activity* fechar. Isso acontece quando o usuário pressiona o botão Home ou botão voltar para sair da aplicação ou, ainda, quando recebe uma ligação no telefone. Ele é chamado para salvar o estado da aplicação.
d) *onClose*() literalmente encerra a execução de uma *activity*. Ele pode ser chamado automaticamente pelo sistema operacional para liberar recursos ou pode ser chamado pela aplicação pelo método *finish*() da classe *activity.*

36. No Android, existem diversos tipos de gerenciadores de *layout*. Alguns podem organizar os componentes na horizontal e vertical, outros podem organizar os componentes em uma tabela com linhas e colunas.

Analise as afirmações Sobre as classes de *layout*, afirma-se que

a) *FrameLayout* é utilizado para organizar os componentes na vertical e horizontal.
b) *LinearLayout* funciona como uma pilha, sendo que uma *view* fica por cima da outra.
c) *RelativeLayout* permite posicionar um componente relativo a outro, por exemplo, abaixo de um componente já existente.
d) *AbsoluteLayout* é o tipo mais comum e simples de *layout*. Pode ser utilizado para organizar os componentes em uma tabela, com linhas e colunas.

37. Um *Fragment* é um componente independente do *Android* que pode ser usado por uma Activity, analise as afirmações abaixo sobre *Fragments:*

I. Fragment é utilizado para dividir uma Activity em várias partes, porém o fragment não tem controle sobre os eventos e não consegue gerenciar seu próprio conteúdo.
II. O Ciclo de vida de um fragment conta com o método onAttach(activity) e é chamado logo depois de um fragment ser associado com a activity, o que acontece assim que a activity infla o *layout* do fragment pela tag <fragment> ou o fragment é adicionado dinamicamente via FragmentTransaction.
III. O método onDetach() é chamado para indicar que o fragment não está mais sendo utilizado e será destruído.
IV. A API do Fragments possui a classe android.app.FragmentTransaction, que é utilizada para adicionar, remover ou substituir os fragments dinamicamente no *layout*.

Estão corretas apenas as afirmativas

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

38. Analise as afirmações a seguir sobre o uso de *Threads, Handler e AsyncTask* em sistemas para *Android*:

I. Nas versões mais atuais do Android, se o código fizer uma operação de I/O na thread principal, o sistema vai lançar a exceção NetWorkOnMainThreadException.

II. No Android, cada aplicação é executada em um único processo e cada processo, por sua vez, tem uma Thread dedicada, a qual é responsável por desenhar e tratar todos os eventos da interface gráfica e é conhecida como Main Thread ou UI Thread.

III. A classe Handler é utilizada para enviar uma mensagem para ser processada pela UI Thread que, geralmente, é um código que vai atualizar a view.

IV. A Main Thread é utilizada para gerenciar todos os eventos e funções da aplicação e a UI Thread é responsável apenas pela interface da aplicação.

Estão corretas apenas as afirmativas

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

39. Analise as afirmações a seguir sobre a classe Handler e AsyncTask do Android:

I. O método onPreExecute() da classe AsyncTask deve ser executado manualmente para dar início ao Thread.

II. O método sendMensage(msg) é um método da classe Handler que envia a mensagem informada para a fila de mensagens para ser processada assim que possível.

III. A classe AsyncTask gerencia internamente as threads e os handlers necessários para atualizar a interface.

IV. A classe AsyncTask contém métodos para atualizar o andamento (progresso) de uma tarefa, por exemplo, o progresso de um download.

Estão corretas apenas as afirmativas

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

40. Sobre a *RecyclerView* do Android, é procedente afirmar que:

a) não possui suporte a animações ao adicionar ou remover elementos da lista.
b) permite alterar o gerenciador de layout para renderizar as views como listas, grids, ou outra forma customizada.
c) não utiliza, diferente do *ListView,* o conceito de *adapters* para preencher o conteúdo da lista.
d) trata os seus eventos de clique, assim como ListView.