1) Qual a definição correta de uma classe abstrata no PHP?

a) Uma classe abstrata não pode ser instanciada por nenhum objeto, podendo apenas ser herdada por outras classes.
b) Uma classe abstrata pode ser instanciada por outras classes desde que não contenha métodos abstratos.
c) Uma classe abstrata deve ser implementada utilizando a palavra reservada *implements* após o nome da classe, conforme exemplo: "`class MinhaClasse implements MinhaClasseAbstrata`".
d) A palavra reservada utilizada para definição de uma classe abstrata é *Abstraction*, conforme segue exemplo "`Abstraction class NomeClasse { /* implementação */ }`".
e) A visibilidade definida nas subclasses deve ser exatamente igual a definida na classe abstrata.

---

2) Pode-se afirmar que os problemas característicos de uma classe com forte acoplamento são:

I) Entendimento prejudicado, pois, para conhecer o real comportamento de uma classe, precisamos conhecer e entender as classes às quais ela está acoplada.
II) As alterações são em geral mais complexas, pois qualquer alteração nas classes acopladas pode afetar (muitas vezes não sabemos como) a classe.
III) Infere diretamente na coesão da classe, quando mais forte o acoplamento mais coeso será o código.
IV) A reusabilidade é prejudicada, uma vez que a classe depende de outras, impedindo que somente a classe seja reutilizada. É preciso sempre carregar as demais classes.

Assinale a alternativa que indica a(s) afirmativa(s) correta(s).

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

---

3) Qual a maneira correta de definir uma interface no PHP?

a) `interface nome_interface { function public metodo(); }`
b) `interface nome_interface { private function metodo() { }; }`
c) `interface nome_interface { function metodo(){ }; }`
d) `class interfaced nome_interface { public metodo(); }`
e) `interface nome_interface { function metodo(); }`

---

4) Em relação a coesão de classes, é correto afirmar:

I) Uma classe altamente coesa é de difícil entendimento.
II) Uma classe pouco coesa tem sua reusabilidade prejudicada.
III) Uma classe altamente coesa facilita sua manutenção.
IV) Uma classe altamente coesa possui mais responsabilidades do que deveria ter.

Assinale a alternativa que apresenta as afirmativas corretas.

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

---

5) No que diz respeito ao controle de exceções no PHP, é correto afirmar:

I) A forma de lançar uma exceção é com a palavra reservada *throw*.
II) Uma forma de gerenciar exceções é a utilização da estrutura *try/catch*.
III) Uma classe *Exception* não pode ser estendida por herança à outra classe.

Assinale a alternativa que indica a(s) afirmativa(s) correta(s).

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

---

6) Analise o trecho de código PHP abaixo e assinale a alternativa que apresenta o valor que será impresso na tela ao fim da execução:

```
$a = "true";
$b = 0;
while ($b <= 10) {
      $b++;
      if ($a === true) break;
      if ($b % 2) continue;
      else $b += 2;
      if ($b % 2) {
            $a = true;
            die($a);
      }
}
echo $b;
```

a) 1
b) 5
c) 11
d) 12
e) true

---

7) Assinale a alternativa que apresenta os nomes de funções PHP que têm como objetivo, respectivamente:
   - Excluir um arquivo;
   - Ler o conteúdo de um arquivo e retornar o valor em *string*;
   - Retornar os índices de um *array*;
   - Ordenar um *array* pelo seu valor, mantendo a associação de chaves.

a) *file_remove, file_get_contents, array_keys, asort.*
b) *file_remove, readfile, array_get_keys, ksort.*
c) *unlink, file_get_contents, array_keys, asort.*
d) *unlink, file_read, array_get_keys, rsort.*
e) *unlink, readfile, array_k, ksort.*

---

8) Analise o trecho de código PHP abaixo e assinale a alternativa que apresenta o valor que será impresso na tela ao fim da execução:

```
class A {
      public $public = null;
      public $_private = null;
      function __construct($v1, $v2){
            $this->public = $v1;
            $this->_private = $v2;
      }
}
function mudaValor($obj, $v){
      $v++;
      $obj->_private += $v;
}
$a = new A("1", "2");
mudaValor($a, $a->public);
echo "{$a->public} - {$a->_private}";
```

a) `1 - 2`
b) `1 - 4`
c) `2 - 2`
d) `2 - 4`
e) `PHP Fatal error: Cannot access private property A::$_private`

A sequência de *queries* SQL abaixo cria a estrutura base de um banco de dados PostgreSQL que será utilizada nas questões 9, 10, 11 e 12:

```
CREATE SCHEMA clube;
CREATE TABLE clube.socio (
  id SERIAL,
  primeiro_nome VARCHAR(40) NOT NULL,
  sobrenome VARCHAR(200) NOT NULL,
  endereco VARCHAR(200) NOT NULL,
  telefone VARCHAR(20) NOT NULL,
  recomendado_por INTEGER,
  data_cadastro TIMESTAMP WITHOUT TIME ZONE NOT NULL,
  CONSTRAINT socio_data_cadastro_key UNIQUE(data_cadastro),
  CONSTRAINT socio_pkey PRIMARY KEY(id)
) WITH (oids = false);
CREATE TABLE clube.instalacao (
  id SERIAL,
  nome VARCHAR(200) NOT NULL,
  custo_por_socio NUMERIC(10,2),
  custo_por_convidado NUMERIC(10,2),
  custo_inicial NUMERIC(10,2),
  CONSTRAINT instalacao_pkey PRIMARY KEY(id)
) WITH (oids = false);
CREATE TABLE clube.agendamento (
  id SERIAL,
  id_socio INTEGER NOT NULL,
  id_instalacao INTEGER NOT NULL,
  inicio TIMESTAMP WITHOUT TIME ZONE NOT NULL,
  fim TIMESTAMP WITHOUT TIME ZONE,
  CONSTRAINT agendamento_pkey PRIMARY KEY(id),
  CONSTRAINT agendamento_fk FOREIGN KEY (id_socio)
    REFERENCES clube.socio(id)
    ON DELETE RESTRICT ON UPDATE CASCADE
    NOT DEFERRABLE,
  CONSTRAINT agendamento_fk1 FOREIGN KEY (id_instalacao)
    REFERENCES clube.instalacao(id)
    ON DELETE RESTRICT ON UPDATE CASCADE
    NOT DEFERRABLE
) WITH (oids = false);
```

---

9) Considerando a SQL abaixo, qual o objetivo da tabela virtual chamada auxiliar?

```
SELECT
      socio.primeiro_nome || ' ' || socio.sobrenome,
      socio2.primeiro_nome || ' ' || socio2.sobrenome,
      socio.data_cadastro
FROM clube.socio
INNER JOIN (
      SELECT socio.recomendado_por, MAX(socio.data_cadastro) data_cadastro
      FROM clube.socio
      WHERE socio.recomendado_por IS NOT NULL
      GROUP BY socio.recomendado_por
      ORDER BY 1
) auxiliar ON auxiliar.data_cadastro = socio.data_cadastro
INNER JOIN clube.socio socio2 ON socio2.id = socio.recomendado_por
```

a) Listar cada sócio, quem recomendou e a data de cadastro.
b) Agrupar por data de cadastro os sócios, devolvendo quem recomendou e a respectiva data.
c) Obter a informação referente a quem recomendou e qual a data de cadastro dos sócios.
d) A *query* possui erros de sintaxe na linha do `ORDER BY` e na ausência da palavra chave `AS`.
e) Buscar quem recomendou um sócio e qual a última data de cadastro de seus recomendados.

---

10) Qual das *queries* SQL abaixo busca corretamente todos os campos dos agendamentos futuros, mostrando também o telefone de quem recomendou o sócio que agendou a instalação, quando existente?

a)
```
SELECT agendamento.*, socio_pai.telefone
FROM clube.agendamento
INNER JOIN clube.socio ON socio.id = agendamento.id_socio
LEFT JOIN clube.socio AS socio_pai ON socio_pai.id = socio.recomendado_por
WHERE NOW() < inicio;
```
b)
```
SELECT *
FROM clube.agendamento
LEFT JOIN clube.socio ON socio.id = agendamento.id_socio
LEFT JOIN clube.socio AS socio_pai ON socio_pai.id = socio.recomendado_por
WHERE NOW() BETWEEN inicio AND fim;
```
c)
```
SELECT agendamento.*, socio_pai.telefone
FROM clube.agendamento
INNER JOIN clube.socio ON socio.id = agendamento.id_socio
INNER JOIN clube.socio AS socio_pai ON socio_pai.id = socio.recomendado_por
WHERE NOW() > fim;
```
d)
```
SELECT agendamento.*, socio.telefone
FROM clube.agendamento
INNER JOIN clube.socio ON socio.id = agendamento.id_socio
WHERE NOW() > fim;
```
e)
```
SELECT agendamento.*, socio.telefone
FROM clube.agendamento
LEFT JOIN clube.socio ON socio.id = agendamento.id_socio
WHERE NOW() < inicio;
```

---

11) Um sócio excêntrico quer alugar todas as instalações do clube para si mesmo, sem nenhum convidado. Qual *query* SQL abaixo apresenta o custo médio total, por instalação, que o sócio terá?

a)
```
SELECT AVG(custo_por_socio + custo_inicial) FROM clube.instalacao WHERE
custo_por_convidado IS NULL;
```
b)
```
SELECT SUM(custo_por_socio) + SUM(custo_inicial) / COUNT(*) FROM clube.instalacao
WHERE custo_por_convidado IS NULL;
```
c)
```
SELECT AVG(COALESCE(custo_inicial, 0) + COALESCE(custo_por_socio, 0)) FROM
clube.instalacao;
```
d)
```
SELECT AVG(COALESCE(custo_por_socio, custo_inicial)) FROM clube.instalacao;
```
e)
```
SELECT AVG(custo_inicial + 0) + AVG(custo_por_socio + 0) FROM clube.instalacao
WHERE TRUE;
```

---

12) Analise a *query* abaixo e assinale a alternativa correta:

```
WITH ag_with AS (
      SELECT * FROM clube.agendamento a
      INNER JOIN clube.instalacao i ON i.id = a.id_instalacao
      RIGHT JOIN clube.socio s ON s.id = a.id_socio
)
SELECT DISTINCT s2.*
FROM ag_with AS aw, clube.socio AS s2
WHERE
      s2.id = aw.id_socio AND
      s2.recomendado_por IS NOT NULL
```

a) Todos os sócios que fizeram agendamento.
b) Todos os sócios que foram recomendados por alguém.
c) Todos os sócios que foram recomendados por alguém e que não fizeram agendamentos.

d) Todos os sócios que foram recomendados por alguém e fizeram agendamentos.
e) A *query* apresenta um erro no *from* do segundo *select*.

---

13) Ao projetar um banco de dados, muitas vezes se faz necessário criar controles para restringir os dados de uma coluna em relação a outras colunas ou linhas, portanto é possível definir restrições em colunas e tabelas. As restrições oferecem todo o controle sobre os dados nas suas tabelas conforme desejar. Considerando um banco de dados PostgreSQL, é correto afirmar que:

I) Uma restrição de chave primária indica que uma coluna ou grupo de colunas pode ser usado como um identificador exclusivo para linhas na tabela. Isso exige que os valores sejam únicos, porém permite valores nulos.
II) Uma restrição de chave estrangeira especifica que os valores em uma coluna (ou um grupo de colunas) devem corresponder aos valores que aparecem em alguma linha de outra tabela.
III) Uma restrição NOT NULL simplesmente especifica que uma coluna não deve assumir o valor nulo.
IV) As restrições exclusivas asseguram que os dados contidos em uma coluna, ou um grupo de colunas, sejam únicos entre todas as linhas na tabela.
V) Uma coluna com restrição UNIQUE automaticamente também tem a propriedade NOT NULL, e, portanto, pode ser identificada como a chave primária da tabela.

Assinale a alternativa que apresenta as afirmativas corretas.

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

---

14) Considerando o banco de dados PostgreSQL, assinale a alternativa que apresenta o nome das instruções usadas para concatenar os elementos de uma matriz, converter uma *string* para data, extrair um pedaço de uma *string* e passar uma *string* para caixa alta, respectivamente:

a) `array_to_string, to_date, substring, upper`
b) `array_join, to_char, substring, upper`
c) `array_to_string, to_char, substr, to_upper`
d) `array_union, to_date, substring, to_upper`
e) `array_cat, to_char, sub_string, upper`

---

15) Analise o trecho de código abaixo e assinale a alternativa que apresenta a saída impressa após sua execução:

```
try {
      var pessoa = {
            ano: 1990,
            calculaIdade: function(anoBase) {
                  var a = anoBase ? anoBase : (new Date().getFullYear());
                  var b = this.ano ? this.ano : 0;
                  return a - b;
            }
      };
      Joao = { ano: 1991 };
      var c = pessoa.calculaIdade.call(Joao, 2017);
      var d = pessoa.calculaIdade.call(null, undefined);
      console.log(c + ", " + d);
} catch(err){
      console.log('Erro ao calcular idades!');
}
```

a) `27, undefined`
b) `26, null`
c) `26, 27`
d) `26, 2017`
e) `Erro ao calcular idades!`

16) Dado o trecho de código HTML abaixo:

```
<body>
<script src="https://code.jquery.com/jquery-3.0.0.min.js"  crossorigin="anonymous">
</script>
<div class="pessoa"><div class="formNome">
<label>Nome:</label><input id="nm_pessoa_id" type="text" name="nm_pessoa">
</div></div>
</body>
```

Qual alternativa abaixo apresenta códigos de comportamento equivalente aos seguintes métodos da jQuery, respectivamente?

- `$("#nm_pessoa_id").hide();`
- `$("#nm_pessoa_id").show();`
- `$('#nm_pessoa_id').attr('disabled','disabled');`

a)
```
$('<label>Nome:</label>').next().attr('display').val('hide');
$("#nm_pessoa_id").not('hide');
$("#nm_pessoa_id").enabled(false);
```

b)
```
$('body').find('input:first').css('display', 'false');
document.getElementsByName('nm_pessoa')[0].style = "element:visible";
$('.formNome>input').style = "element:disabled";
```

c)
```
document.getElementsByName('nm_pessoa')[1].display = 'none';
$("input[name='nm_pessoa']").css('display',"show");
$('pessoa').find('div').next().find('input').attr('readonly', true);
```

d)
```
$("[name='nm_pessoa']").css('display','none');
document.getElementsByName('nm_pessoa')[0].visible = true;
$(".formNome").find('input').attr('activate', false);
```

e)
```
$('input',$('.formNome')).css('display','none');
$('body').find('input:first').css('display',"block");
$(".formNome").find('input').prop('disabled', true);
```

17) Considerando o trecho de código abaixo, marque a alternativa correta:

```
1.  <html><body><nav><ul>
2.  <li><a href="home">Página inicial</a></li><li><a href="sobre">Sobre</a></li>
3.  </ul></nav>
4.  <section id='home'><h1>Home</h1></section>
5.  <section id='sobre' style="display:none"><h1>Sobre</h1></section>
6.  <script src="//ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
7.  <script>
8.  jQuery(document).ready(function () {
9.      jQuery('nav>ul').on('click', 'li', function callback(e){
10.         e.preventDefault();
11.         var link = jQuery(this).find('a').attr('href');
12.         jQuery('section').hide().filter('#'+link).show();
13.     });
14. });
15. </script></body></html>
```

a) O trecho de código `jQuery('nav>ul')` não retornará elementos, visto que não há nenhum elemento `nav` maior que o elemento `ul`.
b) O método `e.preventDefault()` impede a propagação de um evento.
c) O método `e.preventDefault()` é usado em eventos nativos para abortar ações comuns, como cliques em *links*.
d) O trecho de código da linha 9 anexa um evento de *click* aos elementos `nav>ul`, ou seja, quando o elemento `nav` ou `ul` for alvo do clique do *mouse*, a função `callback` será chamada.

e) A função `hide()` presente na linha 12 não existe na jQuery. O correto seria passar um parâmetro booleano `false` para a função `show()`.

---

18) Considerando as funções da jQuery, quais das afirmativas abaixo estão corretas?

I) A função `closest` retorna o elemento mais próximo (iniciando com o elemento atual e subindo pelos pais), que corresponde a um determinado seletor.
II) O método `find` é usado para criar um novo conjunto encapsulado de elementos com base no contexto do conjunto atual e dos seus descendentes.
III) A função `find` é um *alias* para a função `filter`, ou seja, ambas tem o mesmo comportamento.
IV) O método `filter` permite que você filtre o atual conjunto de elementos.

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

---

19) Considere os códigos abaixo:

app.module.ts:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
@NgModule({
  declarations: [AppComponent], imports: [BrowserModule],
  providers: [], bootstrap: [AppComponent]
})
export class AppModule { }
```

app.component.ts:

```
import { Component, OnInit } from '@angular/core';
import { CandidatoService } from './candidato.service';
@Component({
    selector: 'app-root', template: `<h1>{{ titulo }}</h1> <p> {{ nome }} </p> <p>
{{ pontuacao }} </p>`, providers: [CandidatoService]
})
export class AppComponent implements OnInit {
    titulo: string = 'Concurso FURG'; nome: string = "Fulano de tal";
    pontuacao: string | Number;
    constructor(private candidato: CandidatoService) { }
    ngOnInit() { this.pontuacao = '100 pontos'; }
}
```

Em relação aos códigos acima é correto afirmar:

I) A classe `AppComponent` implementa a interface `OnInit`, que exige a implementação da função `ngOnInit`, chamada durante o ciclo de vida de um componente.
II) A definição do argumento candidato como `private` no construtor da classe `AppComponent` gera uma exceção em tempo de transpilação.
III) O decorador de componentes `@component` permite que você marque uma classe como um componente Angular e forneça metadados adicionais que determinem como o componente deve ser processado, instanciado e usado no tempo de execução.

Assinale a alternativa que indica a(s) afirmativa(s) correta(s).

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

---

20) Segundo os conceitos do Angular, qual a forma correta de definir uma ligação de evento de clique a um botão?

a) `<button (click)="salvar()">Salvar</button>`
b) `<button ng-click="salvar()">Salvar</button>`
c) `<button on-click="salvar()">Salvar</button>`
d) `<button ng-onclick="salvar()">Salvar</button>`
e) `<button click("salvar()")>Salvar</button>`

---

21) Considerando os códigos abaixo, ambos no mesmo diretório, para uma aplicação em Node.js:

app.js:

```
var util = require("util"); var api = require("./furg");
api.obtemTelefone("NTI - Núcleo de Tecnologia da Informação", function(data) {
      data.telefones.forEach(function (tel) {
      console.info(
            util.format("Telefone: (%d) %s\nUnidade: %s\nPessoas: %j\n", tel.nr_ddd,
            tel.nr_telefone, tel.unidade, tel.pessoas
));});});
```

furg.js:

```
var https = require("https");
exports.obtemTelefone = function (consulta, callback) {
    var options = {hostname: "api.furg.br",
        path: "/lista_telefonica/Publico/consultaTelefones",
        method: "POST", headers: {"Content-Type": "text/plain"}
    };
    var requisicao = https.request(options, function (res) {
        var body = ""; res.setEncoding("utf8");
        res.on("data", function (chunk) { body += chunk; });
        res.on("end", function () { callback(JSON.parse(body).res); });
    });
    requisicao.write(JSON.stringify({busca_telefone:consulta,limit:1,inicio:0}));
    requisicao.end();
};
```

Saída esperada no console:

```
Telefone: (53) 3233.0000
Unidade: NTI - Núcleo de Tecnologia da Informação
Pessoas: ["JOAOZINHO","MARIAZINHA"]
```

Em relação ao exposto acima, é correto afirmar que:

I) O recurso `util` requerido na primeira linha do `app.js` é um recurso nativo do Node.js.
II) O Node.js possui um sistema de carregamento de módulo simples. Em Node.js, arquivos e módulos estão em correspondência um para um (cada arquivo é tratado como um módulo separado).
III) O comando `util.format`, presente no `app.js`, retorna uma *string* formatada, utilizando marcadores de espaços reservados `%s`, `%d` e `%j`, que representam, respectivamente, *string*, número e *json*, que converte o valor pelos parâmetros correspondentes.
IV) O recurso `https` requerido no arquivo `furg.js` não é um recurso padrão do Node.js. O correto seria fazer o `require` do `http` e passar os parâmetros para que seja realizada uma comunicação `https`.

Assinale a alternativa que apresenta as afirmativas corretas.

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

---

22) Sobre controle de versões de código-fonte, usando versionamento GIT: supondo um diretório remoto válido, recém criado no servidor de controle de versão, quais são os comandos necessários para fazer uma cópia local

de repositório existente, adicionar arquivo ao controle de versão, realizar o primeiro registro de suas alterações e enviar para o servidor remoto, respectivamente?

a) `copy, add, register, send`
b) `clone, add, commit, push`
c) `duplicate, push, save, commit`
d) `pull, push, commit, submit`
e) `init, push, commit, send`

---

23) Segundo a Scrum Guides e os eventos da Scrum, é correto afirmar que:

I) As Sprints são compostas por uma reunião de planejamento da Sprint, reuniões diárias, o trabalho de desenvolvimento, uma revisão da Sprint e a retrospectiva da Sprint.
II) A Reunião Diária é mantida no mesmo horário e local todo dia para reduzir a complexidade. Durante a reunião os membros do time de desenvolvimento esclarecem:
- Qual é o objetivo da Sprint?
- O que pode ser entregue como resultado do incremento da próxima Sprint?
- Como o trabalho necessário para entregar o incremento será realizado?

III) A Revisão da Sprint é executada diariamente durante a sprint, para corrigir falhas no processo.
IV) A Retrospectiva da Sprint é uma oportunidade para o Time Scrum inspecionar a si próprio e criar um plano para melhorias a serem aplicadas na próxima Sprint.

Assinale a alternativa que apresenta as afirmativas corretas.

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

---

24) Segundo a Scrum Guides e o time Scrum, é correto afirmar que:

I) O Time Scrum é composto pelo Product Owner, o Time de Desenvolvimento e o Scrum Master.
II) O Product Owner, ou dono do produto, é o responsável por maximizar o valor do produto e do trabalho do Time de Desenvolvimento.
III) O Time de Desenvolvimento consiste de profissionais que realizam o trabalho de entregar uma versão usável que potencialmente incrementa o produto "Pronto" ao final de cada Sprint. Somente integrantes do Time de Desenvolvimento criam incrementos.
IV) O Scrum Master é responsável por garantir que o Scrum seja entendido e aplicado. O Scrum Master faz isso para garantir que o Time Scrum adira à teoria, a práticas e a regras do Scrum. O Scrum Master é um servo-líder para o Time Scrum.

Assinale a alternativa que apresenta as afirmativas corretas.

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

---

25) Assinale a alternativa que apresenta apenas valores e práticas do Extreme Programming (XP):

a) Simplicidade, Ritmo Sustentável, Código Coletivo e *Sprint.*
b) Jogo do Planejamento, *Design* complexo, Código Padronizado e *Releases* Curtos.
c) Comunicação, Programação em Par, Horas extras e *Releases* Curtos.
d) *Releases* Longos, *Feedback*, Desenvolvimento Guiado pelos Testes e Cliente Presente.
e) Programação em Par, *Refactoring*, Coragem e Integração Contínua.