## PROVA OBJETIVA

**01)** A negação de “eu não gosto de acordar cedo e eu sou solteiro” é:
a) “Eu não gosto de acordar cedo e eu não sou solteiro”
b) “Eu gosto de acordar cedo ou eu não sou solteiro”
c) “Eu gosto de acordar cedo ou eu sou solteiro”
d) “Eu não gosto de acordar cedo ou eu sou solteiro”
e) “Eu gosto de acordar cedo e eu não sou solteiro”

**02)** Considere as seguintes afirmações:
*“Todas as plantas verdes têm clorofila. Algumas coisas que têm clorofila são comestíveis.”*
Tomando por base somente essas afirmações, pode-se concluir que:
a) Alface é comestível.
b) Alface tem clorofila
c) Algumas plantas verdes são comestíveis.
d) Todas as plantas verdes são comestíveis.
e) Todas as plantas que tem clorofila são comestíveis.

**03)** Um programador executa 8 pontos de função quando trabalha 8 horas por dia útil. Um estagiário tem exatamente 50 (cinquenta) por cento da produtividade de um programador. Um cliente especifica 20 (vinte) pontos de função por dia útil e passa esse trabalho para a equipe de desenvolvimento. Desde o início do projeto já se passaram 30 (trinta) dias úteis e a equipe é composta por apenas 1 (um) programador, 2 (dois) estagiários e todos trabalham 6 (seis) horas por dia útil. Sabendo que o cliente ainda especificará pontos de função por mais 15 (quinze) dias úteis, responda em quanto tempo a equipe de desenvolvimento terminará os trabalhos.
a) 45 dias úteis
b) 60 dias úteis
c) 75 dias úteis
d) 90 dias úteis
e) 105 dias úteis

**04)** Ao entrar numa floresta, Alice perdeu a noção dos dias da semana. O leão e o tigre tornaram-se amigos de Alice. Ela sabia que o leão mentia às segundas, terças e quartas e dizia a verdade nos outros dias da semana. Já o tigre mentia às quintas, sextas e sábados e dizia a verdade nos outros dias da semana. Um dia, os dois animais disseram para Alice: “Ontem foi um dos meus dias de mentir”. Qual era o dia da semana?
a) Segunda-feira
b) Terça-feira
c) Sábado
d) Quinta-feira
e) Domingo

**05)** As três principais estruturas de controle usadas na programação estruturada são:

a) Sequência, seleção e repetição.
b) Seleção, desvio incondicional e repetição.
c) Sequência, desvio incondicional e repetição.
d) Seleção, sequência e desvio incondicional.
e) Seleção, repetição e recursividade.

**06)** As afirmações abaixo são relativas à linguagem C.

I - Se p é um ponteiro para x, o comando (*p)++ incrementa x.
II - O operador '&' retorna o endereço da variável que a ele sucede.
III - p[2] equivale a *(p+2).

a) Somente a afirmação I está certa.
b) Somente a afirmação II está certa.
c) As afirmações I e II estão certas.
d) As afirmações II e III estão certas.
e) Todas as afirmações estão certas.

**07)** Sabendo que o ^ (circunflexo) é o operador binário bit-a-bit XOR, assinale a alternativa correta com base no código em linguagem C apresentado a seguir.

```
int main(void){
  int a = 3;
  int b = 2;
  printf("%d %d \n", a, b);
  a = a ^ b;
  b = b ^ a;
  a = a ^ b;
  printf("%d %d \n", a, b);
}
```

a) Após a execução do código acima tem-se a seguinte saída:
3 2
6 4
b) Após a execução do código acima tem-se a seguinte saída:
3 2
3 3
c) Após a execução do código acima tem-se a seguinte saída:
3 2
2 3
d) Após a execução do código acima tem-se a seguinte saída:
3 2
9 6
e) Após a execução do código acima tem-se a seguinte saída:
3 2
3 2

**08)** Assinale a alternativa que mostra a saída apresentada no console após a execução do código em linguagem C abaixo.

```
int func(){
        static int x = 2;
        x += 5;
        return x;
}

int main(){
        printf("%d ", func());
        printf("%d ", func());
        printf("%d ", func());
        return 0;
}
```

a) 2 2 2
b) x x x
c) 5 5 5
d) 7 7 7
e) 7 12 17

**09)** Após a execução do trecho de código abaixo, qual será o valor da variável *q*?

```
int n = 28, d = 8, q;
for (q = 0; n >= d; n = n – d){ q++; }
```

a) 5
b) 4
c) 3
d) A execução desse código resulta num erro.
e) Não é possível determinar o valor de q após o laço.

**10)** Marque a alternativa que apresenta corretamente a saída do console após a execução do código C abaixo.

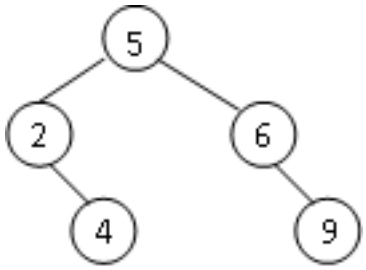
```
int main(int argc, const char * argv[])
{
        int vet[5] = {10, 20, 30, 40, 50};
        int i = 1;

        do
        printf("%d - ", vet[++i]);
        while (i<4);

        return 0;
}
```

a) 20 - 30 - 40 - 50 -
b) 10 - 20 - 30 - 40 -
c) 10 - 20 - 30 -
d) 20 - 30 - 40 -
e) 30 - 40 - 50 -

**11)** Uma versão especializada de uma lista encadeada em que os nós só podem ser inseridos no início da lista e excluídos do final da lista é denominada:
a) Árvore ternária
b) Pilha
c) Árvore binária
d) Fila
e) Tabela hash

**12)** À medida que a string “INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA“ teve seus caracteres lidos da esquerda para a direita os mesmos foram inseridos em uma pilha. Em seguida todos os caracteres foram retirados e concatenados em uma nova string até que a pilha ficasse vazia. A nova string formada foi:
a) “INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA“
b) “TECNOLOGIA E CIÊNCIA EDUCAÇÃO DE FEDERAL INSTITUTO“
c) “AIGOLONCET E AICNÊIC OÃÇACUDE ED LAREDEF OTUTITSNI ”
d) “OTUTITSNI LAREDEF ED OÃÇACUDE AICNÊIC E AIGOLONCET”
e) “EDUCAÇÃO DE FEDERAL INSTITUTO TECNOLOGIA E CIÊNCIA “

**13)** Ao imprimir os valores contidos na árvore abaixo, percorrendo-a em pré-ordem, obtemos a seguinte sequência de valores:




a) 5, 2, 6, 4, 9
b) 2, 4, 5, 6, 9
c) 4, 2, 9, 6, 5
d) 5, 2, 4, 6, 9
e) 9, 6, 5, 4, 2

**14)** Considere o código abaixo para em seguida assinalar a alternativa correta.

```
public class Singleton {

        private static Singleton instance = null;

        private Singleton() {

        }

        public static Singleton getInstance(){
          if(instance == null)
                instance = new Singleton();
          return instance;
        }
}
```

a) O código apresenta erro em tempo de compilação, pois não é possível ter um construtor com modificador de acesso private, tal como apresentado no código acima.
b) O construtor apresentado nesse código pode ser chamado a partir de qualquer outra classe, uma vez que um construtor é obviamente usado para instanciar objetos.
c) A única forma de instanciar um objeto da classe Singleton é através da chamada ao método getInstance(). Logo, podem-se obter quantas instâncias da classe Singleton se desejar.
d) A única forma de instanciar um objeto da classe Singleton é através da chamada ao método getInstance(). Após a execução desse método sempre teremos apenas um objeto instanciado, mesmo que se execute esse método diversas vezes.
e) Pode-se obter uma instância da classe Singleton através da execução do comando: Singleton instance = new Singleton(); .

**15)** Os principais recursos do paradigma de programação orientado a objetos são?
a) Herança, recursividade e encapsulamento.
b) Encapsulamento, ortogonalidade e polimorfismo.
c) Herança, encapsulamento e polimorfismo.
d) Recursividade, herança e polimorfismo.
e) Herança, encapsulamento e ortogonalidade.

**16)** Marque a alternativa que completa corretamente as assertivas I, II, III e IV.

I. Para um artefato a ser modelado com uso de orientação a objetos, é recomendável ter foco nos aspectos principais e ignorar detalhes que são irrelevantes ao problema em questão. A esse processo chamamos de _____.

II. Uma maneira efetiva de tratar um problema complexo é dividi-lo em partes menores. Nesse caso é feita uma _____.

III. Em uma definição de classe _____ descrevem como será o comportamento dos futuros objetos.

IV. Em uma definição de classe _____ armazenarão os possíveis estados dos futuros objetos.

a) I – abstração, II – decomposição, III – operações, IV - variáveis de instância
b) I – decomposição, II – abstração, III – métodos, IV – atributos
c) I – polimorfismo, II – generalização, III – mensagens, IV – estado
d) I – abstração, II – polimorfismo, III – métodos, IV – atributos
e) I – polimorfismo, II - abstração, III – métodos, IV – atributos

**17)** As variáveis declaradas abaixo em Java consomem da memória as seguintes quantidades em *bytes* respectivamente:

```
short var1;
long var2;
float var3;
```

a) 1, 4, 4
b) 1, 4, 8
c) 2, 8, 8
d) 2, 8, 4
e) 2, 4, 8

**18)** Considere as classes *Pessoa* e *App,* apresentadas a seguir, para assinalar a alternativa correta.

```
public class Pessoa {

	private int id;
	private String nome;

	public Pessoa(int id, String nome) {
		this.id = id;
		this.nome = nome;
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getNome() {
		return nome;
	}
	public void setNome(String nome) {
		this.nome = nome;
	}
}
```

```
public class App {

	public static void main(String[] args) {

		Pessoa p1 = new Pessoa(1, "joão");
		Pessoa p2 = new Pessoa(1, "maria");
		Pessoa p3 = p2;
		p2 = p1;
		System.out.printf("%s %s %s",p1.getNome(), p2.getNome(),p3.getNome());

	}
}
```

a) Têm-se 3 referências e 2 instâncias da classe Pessoa, além disto é impresso joão joão joão como resultado.
b) Têm-se 3 referências e 2 instâncias da classe Pessoa, além disto é impresso joão joão maria como resultado.
c) Têm-se 2 referências e 3 instâncias da classe Pessoa, além disto é impresso joão joão maria como resultado.
d) Têm-se 2 referências e 2 instâncias da classe Pessoa, além disto é impresso joão joão maria como resultado.
e) O código apresenta erro em tempo de compilação, pois é necessário ter no mínimo uma instância para cada referência.

**19)** Dado o código Java abaixo, marque a alternativa correta.

```
interface Geometria {
	double getArea  (double rad);
	double toRadios (double grau);
}
interface Conjunto {
	int numConjuntoPartes  (int numElementos);
}
public abstract class Matematica implements Geometria, Conjunto  {
	public int numConjuntoPartes  (int numElementos) {
	return (int) (Math.pow(2, numElementos));
	}
}
```

a) O código compila, mas nenhum objeto pode ser instanciado a partir da classe Matematica.
b) O código compila e não há restrição de instanciação de objetos a partir da classe Matematica.
c) O código não compila, porque a inteface Geometria não foi implementada corretamente na classe Matematica.
d) O código não compila, porque classes abstratas não podem implementar interfaces.
e) O código não compila, porque apenas uma interface pode ser implementada por vez em uma classe.

**20)** Um programador foi designado para projetar uma aplicação na qual Fizzlers são um tipo de Whoosh. Fizzlers também devem ter o comportamento de Oompahs. Adicionalmente, Whooshes têm vários Wingits. Qual código representa esse projeto?

a)
```
class Wingit { }
class Fizzler extends Oompah implements Whoosh { }
interface Whoosh {
    Wingits [ ] w;
}
class Oompah { }
```

b) class Wingit { }
 class Fizzler extends Whoosh implements Oompah { }
 class Whoosh {
     Wingits [ ] w;
 }
 interface Oompah { }

c) class Fizzler { }
 class Wingit extends Fizzler  implements Oompah { }
 interface Whoosh {
     Wingits [ ] w;
 }
interface Oompah { }

d) interface Wingit { }
 class Fizzler extends Whoosh implements Wingit { }
 class Wingit {
         Whoosh [ ] w;
 }
 class Whoosh { }

e) class Fizzler { }
 class Wingit extends Oompah implements Whoosh { }
 interface Whoosh {
         Wingits [ ] w;
 }
 class Oompah { }