Parte teórica. Duração: 30m

Nome: _____ **Código:** _____

Notas:
- Responda às questões seguintes, indicando a opção correta (em maiúsculas)
- Cada resposta errada vale -20% da cotação da pergunta

1. Considere os dois membros-função *calcula1* e *calcula2* de uma classe:

```
class ABC {
public:
  static unsigned int calcula1(...);
  int calcula2(...);
};
```

A. Na função *calcula2* não se pode chamar a função *calcula1* porque *calcula2* é *static* e *calcula1* não
B. Na função *calcula2* não se pode chamar a função *calcula1* porque *calcula2* tem um tipo de retorno diferente do tipo de retorno de *calcula1*
C. Pelo facto de haver um membro-função *static* a classe é abstrata.
D. Durante a execução do programa, existe uma implementação de *calcula1* em memória mesmo que não existam objetos dessa classe.
E. Nenhuma das possibilidades anteriores

Resposta: _____

2. Considere as declarações:

```
float soma(float x, float y);
float soma(float x, float y, float z);
int soma(int x, int y);
```

para fazer *overloading* da função soma. É verdade que as declarações:

A. Estão erradas porque a segunda declaração tem mais um argumento que as outras
B. São válidas e representam um exemplo correto de *overloading* de funções
C. Estão erradas porque no *overloading* de funções é obrigatório as declarações terem o mesmo nome só se podendo variar o número de argumentos e não o seu tipo
D. Estão erradas porque no *overloading* de funções é obrigatório as declarações terem o mesmo nome só se podendo variar o tipo dos argumentos e não o seu número
E. Nenhuma das possibilidades anteriores

Resposta: _____

3. Numa hierarquia de classes C++ é verdade que se houver um membro-função virtual puro na classe base:

A. Obrigatoriamente todas as classes derivadas são abstratas
B. Uma classe derivada que não implemente essa função é abstrata
C. O programa só pode criar um objeto dessa classe base
D. A implementação desse membro-função tem que ser feita num ficheiro separado
E. Nenhuma das possibilidades anteriores

Resposta: _____

4. Considere a seguinte definição da classe *Classe1*:

```
class Classe1 {
private:
  int info;
public:
  Classe1(int info);
};
```

A. A declaração *Classe1 c;* dá erro de compilação
B. A declaração *Classe1 c;* não dá erro de compilação, pois o C++ fornece um construtor sem argumentos por omissão
C. A definição da classe não compila corretamente porque não tem um membro-função para acesso ao membro-dado
D. A declaração *Classe1 c;* dá erro de compilação porque *info* não pode ser usado em simultâneo como membro-dado e como parâmetro do construtor
E. Nenhuma das possibilidades anteriores

Resposta: _____

5. A classe **Mestrando** é uma classe derivada da classe **Estudante**, que é uma classe derivada da classe **Pessoa**. Considere as seguintes declarações de variáveis:

```
Pessoa *p=new Pessoa(); Estudante *e=new Estudante(); Mestrando *m =new Mestrando();
```

Quais das seguintes atribuições estão corretas?

```
I. p = m;          II. p = new Mestrando();      III. m = new Estudante();
IV. m = p;         V. e = new Pessoa();
```

A. III e IV
B. I e IV
C. I e II
D. II, III e V
E. Nenhuma das possibilidades anteriores

Resposta: _____

6. Qual das seguintes afirmações é verdadeira, relativa a um membro-função abstrato herdado por **ClasseX**?

A. Tem de ser obrigatoriamente definido em **ClasseX** para que **ClasseX** possa ser instanciada
B. Não permite que **ClasseX** seja uma classe base
C. Obriga a que **ClasseX** seja sempre uma classe abstrata
D. Redefine qualquer membro-função em **ClasseX** com o mesmo nome
E. Nenhuma das possibilidades anteriores

Resposta: _____

Nome: _____ **Código:** _____

7. Considere as seguintes declarações:

```
class A;    class B;    class C: public A;    class D: public A, public B;
```

A. Não é possível haver herança múltipla em C++, portanto a declaração da classe **D** está errada
B. A declaração da classe **D** só será possível se a classe **A** for puramente abstracta
C. A declaração da classe **D** só será possível se a classe **B** for puramente abstracta
D. A classe **D** só poderá implementar métodos estáticos, herdados de **A** e de **B**
E. Nenhuma das possibilidades anteriores

Resposta: _____

8. Todas as classes declaradas abaixo (`A, B, C` e `D`) implementam um método `string print()`, que retorna uma *string* com informação específica sobre cada uma das classes.

```
class A;    class B;    class C: public A;    class D: public A, public B;
```

A. O método `print()`, implementado em **D**, é um polimorfismo do mesmo método, herdado de **A** e **B** simultaneamente
B. Se o método `print()` for declarado como *virtual* em **A**, então **C** e **D** são classes polimórficas de **A**, por reimplementarem diferentes comportamentos para `print()`
C. As classes **A** e **B** são consideradas polimórficas entre si, pois implementam diferentes comportamentos para o mesmo método *print()*
D. **D** não pode derivar de **A** e **B**, pois estas são polimórficas entre si
E. Nenhuma das possibilidades anteriores

Resposta: _____

9. Qual das seguintes afirmações é verdadeira?
A. Um membro-dado estático tem de ser obrigatoriamente inicializado no construtor
B. Um membro-dado estático não pode ser constante
C. Apenas membros-função estáticos podem aceder a um membro-dado estático
D. Os membros-função estáticos não podem aceder a membros-dado não estáticos
E. Nenhuma das possibilidades anteriores

Resposta: _____

10. Considere as seguintes declarações:

```
class Base;        class Derivada: protected Base;
```

A. A classe **Derivada** herdará apenas os membros protegidos da classe **Base**
B. Os membros públicos da classe **Base** passarão a ser protegidos na classe **Derivada**
C. Os membros privados da classe **Base** passarão a ser protegidos na classe **Derivada**
D. A classe **Derivada** passará a ser protegida e apenas acessível por métodos *get* e *set*
E. Nenhuma das possibilidades anteriores

Resposta: _____