

		<b>Escola de Engenharia da Universidade do Minho</b> Mestrado Integrado em Eng. Electrónica Industrial e Computadores <b>Complementos de Programação de Computadores</b>						<b>2015/2016</b> <b>MIEEIC</b> <b>(1º Ano)</b> <b>2º Sem</b>	
<b>Docentes: Luís Paulo Reis, Isabel Moura</b>									
<b>Exame Nº1 - Época Normal - Data 06/04/2016, Duração 1h45m+10min (com consulta)</b>									

Nome: \_\_\_\_\_ Nº Aluno: \_\_\_\_\_

Responda a todas as questões utilizando exclusivamente as folhas agrafadas fornecidas. **Não escreva nada em qualquer outra folha durante a realização do exame.** Suponha que foram realizados as inclusões das bibliotecas necessárias (`#include`) e a instrução: `"using namespace std;"`. Caso considere que a informação fornecida é insuficiente indique os pressupostos que assumiu na sua folha de teste.

Responda a 12 das 14 questões, preenchendo a tabela com a **opção correcta (em maiúsculas)** (Correcto: x Val / Errado: -x/3 Val).

1	2	3	4	5	6	7	8	9	10	11	12	13	14

## GRUPO I (5.0 Valores)

1) Suponha a seguinte classe CPonto:

```
1 class CPonto {
2     double x; double y;
3     public:
4         CPonto();
5         CPonto(double x, double y);
6         CPonto(const CPonto& out);
7         ~CPonto();
8         void Transla(double dx, double dy);
9         void Escala(double fx, double fy);
10        double DistAte(const CPonto& out);
11 };
```

Qual das seguintes alternativas é verdadeira:

- A) Antes da linha 2 deveria estar a declaração *private* porque senão os dados são públicos.
- B) A linha 2 declara dois construtores da classe CPonto (double x e double y).
- C) As linhas 4 a 6 contêm a declaração das funções da classe.
- D) A linha 11 tem um erro (ponto e vírgula no final que nunca se usa depois de uma chaveta)
- E) Nenhuma das anteriores.

2) Supondo o seguinte código indique o que é escrito no écran antes de ser escrita a palavra "fim".

```
1. class CBase {
2.     public:
3.         CBase() {cout << "A";}
4.         ~CBase() {cout << "B";}
5. };
6. CBase x1;
7.
8. int main()
9. {
10.     { CBase x2; }
11.     cout << "B";
12.     CBase x3;
13.     cout << "fim";
14.     return 0;
15. }
```

- A) AAABBBB B) AABBBB C) AABBA D) AAABBB
- E) Nenhuma das Anteriores

3) A classe CEstudante é uma classe derivada da classe CPessoa. Ambas as classes possuem a sua implementação do membro-função público `virtual void funcaoMisterio1()`. Considere o seguinte segmento de código:

```
CPessoa *ap = new CEstudante("Paulo", "MIEEIC");
ap-> funcaoMisterio1();
```

Que implementação do membro-função `funcaoMisterio1()` é invocada na segunda linha?

- A) A implementação existente na classe CPessoa
- B) A implementação existente na classe CEstudante
- C) CPessoa e CEstudante não podem implementar membros-função virtuais com igual assinatura
- D) A linha 1 está errada, a variável `ap` é inicializada de forma incorreta
- E) Nenhuma das anteriores

4) Na linguagem C++, qual das seguintes afirmações é verdadeira:

- A. Todos os operadores existentes podem ser redefinidos
- B. Apenas os operadores binários podem ser redefinidos
- C. Na redefinição de um operador, é possível ter um objeto como operando
- D. Na redefinição de um operador é possível utilizar uma sintaxe diferente da do operador original
- E. Nenhuma das possibilidades anteriores

5) O que escreve no écran o seguinte código?

```
int a = 200, b = 100, *p = &a, *q = &b;
*p = *q; a = b;
cout << *p << *q;
```

- A) 100100 B) 200200 C) 100200 D) 200100
- E) Nenhuma das anteriores

6) Como declarar uma variável `ap` que seja um apontador constante para um valor inteiro?

- A) `int * const ap;`
- B) `int const * const ap;`
- C) `int const *ap;`
- D) `int *ap;`
- E) Nenhuma das anteriores

7) Considere o seguinte fragmento de código:

```
class A {
    friend class B;
...
};
```

- A) O qualificador `friend` produz o mesmo efeito que tornar B uma subclasse de A
- B) O qualificador `friend` permite a B acesso a todos os membros de A
- C) O qualificador `friend` permite a B acesso apenas aos membros-função de A
- D) O qualificador `friend` não produz qualquer efeito neste caso
- E) Nenhuma das anteriores

8) Executando o seguinte código, o que é escrito no écran?

```
void x(int a=3, int b=4){ cout << a << b; }
int main(){
    x(1);
    x(3, 4);
    return 0;
}
```

A) 1143 B) 1443 C) 1134 D) 4143

E) Nenhuma das anteriores

9) Supondo a classe `CAIuno`, indique qual das alternativas é verdadeira:

```
1 class CAIuno {
2     int numAluno;
3     string nome;
4     double nota;
5 public:
6     CAIuno();
7     CAIuno(int, string);
8     ~CAIuno();
9     double getNota() const;
10    int getNumAluno() const;
11    void setNumAluno(int);
12    string getNome() const;
13 };
```

A) Os dados da classe são privados.

B) A linha 6 tem um erro (o construtor tem de ter parâmetros).

C) A linha 11 tem um erro (falta o *const* antes do ; tal como sucede com as linhas 9, 10 e 12).

D) A linha 7 tem um erro (o construtor já estava declarado).

E) Nenhuma das anteriores.

10) 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

11) 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) Estão corretos: III e IV

B) Estão corretos: I e IV

C) Estão corretos: I e II

D) Estão corretos: II, III e V

E) Nenhuma das possibilidades anteriores

12) Quando existe uma implementação de um destrutor, a intenção mais típica é:

A) Inicializar os membros da classe

B) Libertar explicitamente memória que de outro modo ficaria inacessível

C) Respeitar a regra do C++ que diz que é obrigatória uma implementação do destrutor

D) Ter tantos construtores como destrutores

E) Nenhuma das possibilidades anteriores

13) Considere a seguinte declaração das classes `Base` e `Derivada`:

```
class Base {
public:
    virtual void foo(){ cout<<"Base"; }
    void foo(int i) { cout<<"Base "<< i; }
};
class Derivada: public Base {
public:
    void foo(){ cout<<"Derivada"; }
    void foo(int i) { cout<<"Derivada "<< i; }
};
```

É correto afirmar:

A) Na classe `Base`, `void foo(int i)` é um overload da função `void foo()`

B) Não é necessário existir a implementação da função `void foo()` na classe `Derivada`, pois é virtual na classe `Base`

C) A função `void foo()` na classe `Derivada` está errada, pois deveria ser declarada virtual

D) A função `void foo()` na classe `Derivada` é um overload da sua implementação na classe `Base`

E) Nenhuma das possibilidades anteriores

14) Considere a declaração das classes `Base` e `Derivada` da questão anterior.

A) Na execução do código `Derivada d; d.foo();` é escrito no monitor "Base"

B) Na execução do código `Base *d=new Derivada(); d->foo();` é escrito no monitor "Base"

C) Na execução do código `Derivada d; d.foo(3);` é escrito no monitor "Base3"

D) Na execução do código `Base *d=new Derivada(); d->foo(3);` é escrito no monitor "Base 3"

E) Nenhuma das possibilidades anteriores

## GRUPO II (11.0 Valores)

Considere uma aplicação para gestão dos correios (CTT). Uma central de correio (classe CCentral) gere a correspondência de várias agências (classe CAgencia). Uma agência de correios tem um nome e uma lista com toda a correspondência recebida à espera de ser expedida. A correspondência (classe CCorrespondencia) pode ser de dois tipos: correio normal (classe CCorreioNormal); ou EMS (Express Mail Service – classe CEMS). A correspondência CorreioNormal tem um preço que é definido na altura da criação de um objecto e é guardado no membro-dado preco. A correspondência do tipo EMS tem um preço que é 5 vezes o seu peso. As classes CCentral, CAgencia, CCorrespondencia, CCorreioNormal e CEMS estão parcialmente definidas a seguir. **Nota:** as classes apresentadas abaixo estão incompletas. Pode adicionar os métodos auxiliares que considerar necessários para a resolução do problema. Suponha que foram incluídos todos os #include necessários e using namespace std;.

```
class CCorrespondencia {
protected:
    string destino;
    int prioridade;
public:
    CCorrespondencia(){ destino=""; prioridade=0;}
    virtual int getPreco()=0;
// ...
};
class CCorreioNormal: public CCorrespondencia {
    int preco;
public:
    CCorreioNormal(){ preco=0;}
    int getPreco();
// ...
};
class CEMS: public CCorrespondencia {
    int peso;
public:
    CEMS() { peso=0;}
    int getPreco();
// ...
};
class CAgencia {
    string nome;
    vector<CCorrespondencia*> cartas;
public:
    CAgencia(string no) { nome=no; };
    int adicionaCarta(CCorrespondencia *c);
    bool estaOrdenado() const;
// ...
};
class CCentral {
    int ID;
    vector<CAgencia> agencias;
public:
    CCentral(int ident) {ID=ident; };
// ...
};};
```

2.1) Implemente novos construtores para as classes CCorrespondencia, CCorreioNormal e CEMS que inicializem as suas variáveis com parâmetros passados no construtor:

```
CCorrespondencia(string de, int pri);
CCorreioNormal(int pr, string de, int pri)
CEMS(int pe, string de, int pri);
```

2.2) Implemente as funções int getPreco() das classes CCorreioNormal e CEMS.

2.3) Implemente o membro função da classe CAgencia que adiciona uma carta à agência, retornando o número total de cartas:

```
int CAgencia::adicionaCarta(CCorrespondencia *c);
```

2.4) Implemente uma função booleana para a classe CAgencia que verifica se, numa lista de correspondências, estas estão ordenadas por ordem crescente de preço:

```
bool CAgencia::estaOrdenado() const.
```

2.5) A central de correios recolhe das agências a correspondência para ser expedida. O processamento é feito por regiões. Implemente na classe `CAgencia` um membro-função que recebe o nome de um destino e devolve um vector com toda a correspondência para esse destino: `vector <CCorrespondencia*> recolhaCorr(string destino);` Esta função deve também remover do vector cartas da agência a correspondência que satisfaz as condições pedidas.

2.6) Na classe `CAgencia`, implemente um membro-função `int CAgencia::retornaValor()` que retorne o valor total em correspondência que a agência tem para expedir.

2.7) Na classe `CCentral`, implemente um membro-função `void CCentral::ordenaAgencias()` que ordene o vector de agências da central por ordem decrescente do valor da sua correspondência. Nota: Mesmo que não a tenha implementado, utilize a função da alínea anterior.

2.8 Implemente um operador `+` na classe `CAgencia` que permita juntar duas agências. O nome da agência resultado deverá ser a concatenação dos dois nomes e a correspondência da agência resultado deverá ser o conjunto de correspondência de ambas as agências.

2.9) Implemente uma função `main()` (programa principal) que lhe permita testar as alíneas anteriores.

### GRUPO III (4.0 Valores)

Pretende-se guardar informação sobre uma cadeia de supermercados de venda de bebidas:

- Existem um conjunto de lojas/supermercados de venda de bebidas, cada qual situado numa dada cidade, com uma dada dimensão (em m2) e com um determinado gerente (empregado da loja).
- Cada loja tem também um conjunto de empregados dos quais interessa saber o nome, nº CC, salário e data de contratação.
- A cadeia de supermercados deseja ainda ter informação das distâncias entre cada par de lojas.
- As lojas têm ainda um conjunto de produtos de diversos tipos, interessando saber para cada produto o seu código e nome.
- Os produtos podem ser de três tipos: pipos de líquidos, dos quais interessa saber o conteúdo actual e a capacidade máxima (ambos em litros); garrafas (das quais interessa saber o número em stock e a capacidade - em litros); acessórios (dos quais interessa saber a função e o prazo de garantia - em meses).
- Para todos os produtos interessa saber o preço por unidade.
- Vendas. Cada venda precisa de registar o número de contribuinte do cliente, produtos e respectivas quantidades que adquiriu, empregado que o atendeu e preço total da venda.

Defina um conjunto de classes que na sua opinião melhor descreve o cenário acima, especificando os dados, construtores e destrutores (se necessário) de cada classe. Implemente ainda métodos get e set, exemplificativos, para uma única classe à sua escolha Justifique as suas escolhas. *Nota: Não é necessário implementar qualquer método, simplesmente definir os ficheiros \*.h das classes respetivas.*