Universidade do Minho	**Escola de Engenharia da Universidade do Minho** Mestrado Integrado em Eng. Electrónica Industrial e Computadores Complementos de Programação de Computadores	**2013/2014** **MIEEIC** **(1º Ano)** **2º Sem**
Docentes: Luis Paulo Reis, Pedro Pimenta e C. Filipe Portela		
Teste Nº1 - Época Normal - Data 09/04/2014, Duração 1h45m+10min (com consulta)		

Nome:_____ Nº Aluno:_____

Responda às seguintes questões, preenchendo a tabela com a **opção correcta (em maiúsculas)** **(Correcto:x Val / Errado: -x/3 Val).** Suponha que foram realizados as inclusões das bibliotecas necessárias e "using namespace std;".

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

GRUPO I (6.0 Valores)

1) Suponha a seguinte classe `CPonto`:

```
1  class CPonto {
2   protected:
3     double x, y;
4   public:
5     CPonto();
6     CPonto(double x, double y);
7     CPonto(const CPonto& out);
8     ~CPonto();
9     void Transla(double dx, double dy);
10    void Escala(double fx, double fy);
11    double DistAte(const CPonto& out);
12 };
```

Qual das seguintes alternativas é verdadeira:

A) A linha 12 tem um erro (ponto e virgula no final que nunca se usa depois de uma chaveta)
B) A linha 2 tem um erro (deveria ser *private* e não *protected*).
C) A linha 3 declara dois membros função da classe `CPonto` (uma função `double x` e outra `double y`).
D) As linhas 5 a 7 contêm a declaração dos métodos da classe.
E) Nenhuma das anteriores.

2) Como declarar uma variável x que seja um apontador para um valor constante inteiro?

A) int * const x;
B) int const * const x;
C) int const *x;
D) int *x;
E) Nenhuma das anteriores

3) Supondo o seguinte código indique o que é escrito no écran.

```
1.  class Base {
2.    public:
3.      int m;
4.      Base(int n=0): m(n) {
5.          cout << "Base! ";}
6.  };
7.  class Derived: public Base {
8.    public:
9.      double d;
10.     Derived(double de = 0.0):d(de){
11.         cout << "Derived! ";}
12. };
13. int main()
14. {
15.   cout << "Base ";
16.   Base cBase;
17.   cout << "Derived ";
18.   Derived cDerived;
19.   return 0;
20. }
```

A) Base Base! Derived Derived!
B) Base Base! Derived Base! Derived!
C) Base Derived Base! Derived!
D) Base Derived Derived! Base!
E) Nenhuma das Anteriores

4) Supondo seguinte código indique o que é escrito no ecrán.

```
1.  #include <string>
2.  int main (){
3.     string str ("Test String");
4.     for(string::iterator it = str.begin();
           it != 5; ++it)
5.         cout << *it;
6.     return 0;
7.  }
```

A) Test
B) String
C) Test String
D) Nada pois dá erro na compilação
E) Nenhuma das anteriores

5) O que acontece no final da execução do seguinte código?

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

A) b fica com o valor de a!
B) p aponta agora para b!
C) a fica com o valor de b!
D) q aponta agora para a!
E) Nenhuma das anteriores

6) Dada a seguinte declaração, escolha a opção correta

```
string* x, y;
```

A) x é um apontador para uma string e y é uma string
B) y é um apontador para uma string e x é uma string
C) x e y são apontadores para strings
D) x e y são strings (do tipo string STL)
E) Nenhuma das anteriores

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

```
1.  void Values(int n1, int n2 = 10){
2.     cout << "1st val: " << n1 << " ! ";
3.     cout << "2nd val: " << n2 << " ! ";
4.  }
5.  int main(){
6.     Values(1);
7.     Values(3, 4);
8.     return 0;
9.  }
```

A) 1st val: 1 ! 2nd val: 10 ! 1st val: 3 ! 2nd val: 4 !
B) 1st val: 1 ! 2nd val: 10 ! 1st val: 3 ! 2nd val: 10 !
C) 1st val: 0 ! 2nd val: 1 ! 1st val: 3 ! 2nd val: 4 !
D) Nada pois dá erro de compilação
E) Nenhuma das anteriores

8) Considere o seguinte fragmento de código C++ e indique a resposta correta.

```
1  class CRectangle {
2    private:
3       float width, height;
4    public:
5       void set_values (float a, float b) {
6            width=a; height=b; }
7       friend class CSquare;
8  };
9  class CSquare {
10   private:
11      float side;
12   public:
13     void convert_rect(CRectangle a) {
14         side = (a.width+a.height)/2.0; }
15 };
```

A) A função `friend class` (linha 7) não está corretamente declarada
B) A função `convert_rect` (linhas 13 a 14) não pode usar os membros-dado `width` e `height`, da classe `CRectangle`, (que são privados) por isso está erradamente declarada
C) O programa está correto e permite converter retângulos em quadrados
D) O programa não está correto pois falta a declaração `friend class CRectangle;` na classe `CSquare`
E) Nenhuma das anteriores

9) Para que membros de uma classe possam ser somente acessíveis pela própria classe e pelas subclasses respetivas (i.e. classes derivadas):
A) Devem estar declarados na zona `public` da classe
B) Devem estar declarados na zona `protected` da classe
C) Devem estar fora de qualquer zona da classe (i.e. declarados entre a classe e a sub-classe de modo a estarem acessíveis a ambas);
D) Devem ser constantes globais;
E) Nenhuma das anteriores.

10) Considere a classe base `CPolygon` e a classe derivada `CRectangle` declaradas no seguinte fragmento de código:

```
1   class CPolygon {
2     protected:
3       int width, height;
4     public:
5       void set_values (int a, int b) {
            width=a; height=b; }
6   };
7   class CRectangle: public CPolygon {
8     public:
9       int area () { return (width * height); }
10  };
11  int main () {
12    CRectangle *rect = new CRectangle;
13    CPolygon *ppoly1 = rect;
14    ppoly1->set_values (4,5);
15    cout << rect->area() << endl;
16    return 0;
17  }
```

A) O programa escreve 20 no écran;
B) O programa escreve 0 no écran;
C) Para ficar correto é necessário alterar a instrução: `'ppoly1->set_values(4,5);'` para a instrução: `'ppoly1.set_values(4,5)';`
D) Para o programa ficar correto é necessário alterar a instrução `'ppoly1->set_values(4,5);'` para a instrução: `'rect->set_values(4,5);'`
E) Para o programa ficar correto é necessário alterar a instrução: `'cout << rect->area() << endl;'` para: `'cout << rect.area() << endl;'`

11) Executando o seguinte código indique o que é escrito no écran.

```
1  class sample{
2    public:
3       virtual void example() = 0;
4  };
5  class Ex1:public sample {
6     public:
7        void example() { cout << "Buga!"; }
8  };
9  class Ex2:public sample {
10    public:
11       void example() { cout << "Uga!"; }
12 };
13 int main() {
14    sample* arra[2];
15    Ex1 e1;  Ex2 e2;
16    arra[0]=&e1; arra[1]=&e2;
17    arra[0]->example();
18    arra[1]->example();
19 }
```

A) Buga!
B) Buga!Uga!
C) Uga! Buga!
D) Uga!
E) Nenhum dos Anteriores

12) Quando se usa um `vector` de inteiros em C++ (usando `#include <vector>`) declarado como `'vector <int> v;'` e se pretende adicionar um dado elemento ao `vector`:
A) Tem que se realizar, quando se adiciona um elemento, um `resize(v)` para garantir que existe memória para inserção de novos elementos (memória dinâmica);
B) Pode-se usar o operador "<<" para colocar esses mesmos elementos no `vector`, i.e.: `'v << elemento;'`
C) Pode-se utilizar a instrução `'v.push_back(elemento)'`.
D) Pode-se usar o operador ">>" para colocar o `elemento` no `vector`, fazendo: `'elemento >> v;'`
E) Nenhuma das anteriores.

13) Suponha que deseja representar publicações que podem ser livros ou revistas.
A) Devem ser utilizadas três classes distintas mas não faz sentido utilizar o conceito de herança.
B) Podem ser utilizadas classes: publicação, livro e revista, sendo que livro e revista podem ser classes derivadas de publicação.
C) O mais lógico será utilizar uma classe publicação derivada de livro e de revista.
D) Dado que as pessoas compram todos os tipos de publicações, o melhor será representar tudo na mesma classe.
E) Nenhuma das anteriores.

14) Uma classe pode herdar dados e métodos de mais do que uma classe base?
A) Pode herdar dados e métodos mas unicamente de uma única classe base
B) Pode herdar dados e métodos de uma ou então duas classes base
C) Pode herdar dados e métodos de várias classes base
D) As classes herdam dados mas das classes derivadas
E) Nenhuma das anteriores

15) Ao declarar um Vector (da classe Vector) o que é opcional na sua declaração?
A) O Tipo
B) O Nome dado ao Vector
C) A inclusão da palavra Vector
D) O número de elementos do Vector
E) Nenhuma das Anteriores

GRUPO II (10.0 Valores)

Considere que num casino existem dois tipos de jogos: ***blackjack*** e ***poker***. Quando um jogador entra num casino é-lhe atribuído um número interno (único e sequencial) e um determinado nº de fichas. Cada jogo está associado a uma mesa. Em cada jogo existe um conjunto de jogadores. **Nota:** as classes **Casino**, **Jogo** e **Jogador** estão incompletas. Pode adicionar os métodos auxiliares que considerar necessários para a resolução do problema. Suponha que foram incluídos todos os #include necessários.

```
class Casino {
   vector<Jogo *> mesas;
};

class Jogo {
  protected:
   string nomeJogo;
   vector<Jogador> jogadores;
        // posicao no vector indica vez de jogada
public:
   Jogo(string nm) { nomeJogo=nm; }
   string getNomeJogo() const { return nomeJogo; }
   vector<Jogador>  getJogadores()  const  {  return
jogadores; }
};

class Jogador {
   string nomeJogador;
   int codigo;
   int numFichas;
public:
   Jogador (int cod, int nf, string nomeJ)
      { codigo=cod; numFichas=nf; nomeJogador=nomeJ; }
   int getCodigo() const { return codigo; }
   string getNomeJogador() const {return nomeJogador;}
};
```

2.1) Implemente na classe **Jogador** o método `getNumFichas`

2.2) Implemente as classes **BlackJack** e **Poker** como subclasses de **Jogo**. Implemente os respetivos construtores. Considere que a classe **BlackJack** possui o membro-dado *dealer*, que é uma string que indica o nome do jogador que tem o papel de dealer (distribui as cartas) nesse jogo. Considere que a classe **Poker** possui o membro-dado *apostaMin*, que é um inteiro indicativo do nº mínimo de fichas que qualquer jogador é obrigado a apostar nesse jogo.

2.3) Implemente na classe **Jogador** o método: *void imprimeJogadorInfo()*. Este método imprime no standard output (monitor) informação sobre o jogador incluindo o código, nome e fichas.

2.4) Implemente na classe **Casino** o método: *void imprimeInfo()*. Este método imprime no standard output (monitor) informação sobre todos os jogos existentes, nomeadamente: o nome do jogo, informação própria do jogo (qual o *dealer* se blackjack, ou qual a *aposta mínima* se poker) e o nome dos jogadores que estão nesse jogo.

2.5) Implemente na classe **Jogo** o método *bool eliminaJogador(string nomeJ)*. Este método procura o jogador de nome *nomeJ* no jogo. Se existir, elimina-o, escreve no ecran uma mensagem "Jogador Nome eliminado" e retorna *true*. Se não existir, escreve "Jogador Nome inexistente" e retorna *false*.

2.6) Implemente na classe **Jogador** o operador ++ *Jogador operator++().* Suponha que incrementar um jogador significa simplesmente duplicar as suas fichas!

2.7) Implemente na classe **Jogo** o operador + *Jogo operator + (const Jogo &jogo2).* Dois jogos só podem ser somados se tiverem o mesmo nome. A soma de dois jogos é um outro jogo cujos jogadores são os existentes nos dois jogos (não precisa de eliminar repetidos).

2.8) Implemente na classe **Casino** o método: *void ordenaMesas().* Este método ordena o vector *mesas*, crescentemente pelo número de jogadores presentes em cada mesa.

GRUPO III (4.0 Valores)

Pretende-se guardar informação sobre uma universidade.

- A universidade é composta por um conjunto de escolas, cada qual com diversos departamentos. Interessa saber os respetivos códigos, nomes, morada, telefone, responsáveis (reitor/presidente/diretor).
- A universidade tem um conjunto de cursos que podem ser licenciaturas, mestrados, mestrados integrados ou doutoramentos, embora seja suposto que a informação de todos os cursos é idêntica: código, nome, plano curricular (disciplinas, anos, carga horária e respetivos ECTS), departamento(s) envolvido(s) e diretor de curso.
- Existem na universidade funcionários, alunos e docentes. Todos eles têm nome, morada, telefone, data de nascimento e código. Os alunos têm também o curso em que estão matriculados e o conjunto de disciplinas a que se inscreveram em cada ano letivo e respetivas notas/resultados. Os docentes têm também o conjunto de disciplinas que lecionam.
- Cada disciplina, tem um código, é lecionada por um conjunto de docentes sendo um o seu responsável, tem um conjunto de alunos e tem ainda o ano, ECTS, carga horária e programa.

Defina uma hierarquia de classes que na sua opinião melhor descreve o cenário acima, especificando os dados, construtores e destrutures (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.*