Recibo da Frequência de Programação Orientada para Objectos (IGE e ETI), 2003/7/1 **2º semestre de 2002/2003, ISCTE**
Nome do aluno: .....
Número do aluno: .....
Assinatura do docente: .....

Frequência

Programação Orientada para Objectos

IGE e ETI

2003/7/1 - 2º semestre de 2002/2003

ISCTE

Notas:

- Leia atentamente todo o enunciado (incluindo estas notas) antes de começar a prova.
- Esta frequência tem três questões.
- Duração: 3:30h (três horas e trinta minutos).
- A prova deve ser efectuada sem consulta.
- Sobre a sua secretária só pode estar o enunciado, uma caneta e um documento de identificação (coloque pastas, telemóveis, estojos e quaisquer outros objectos sobre a cadeira ao seu lado).
- Desligue o telemóvel.
- Deve responder a todas as questões no próprio enunciado, nos espaços reservados.
- Use as páginas em branco ao longo do enunciado para rascunho.
- Não destaque as folhas do enunciado (nem mesmo as de rascunho).
- Preencha o seu nome e número no recibo (não o destaque). Este ser-lhe-á entregue, assinado por um docente, depois da prova.
- Só um recibo assinado servirá de prova de que entregou a resolução deste enunciado.
- As cotações encontram-se abaixo de cada alínea (total de 20 valores).
- As notas serão afixadas até às 19:00h de segunda-feira, dia 7 de Julho de 2003.
- A revisão de provas terá lugar das 9:40h às 12:30h de terça-feira, dia 8 de Julho de 2003, no gabinete D6.19.
- Boa sorte.

Questão 1

Assinale com V (Verdadeiro) as expressões que estão correctas e com F (Falso) as que estão incorrectas.

Os quadrados podem ser deixados em branco ou ser preenchidos com V ou F.

Em geral as alíneas podem ter zero ou mais respostas correctas. Cada resposta correctamente assinalada vale 0,5 valores. Respostas incorrectas correspondem a um desconto de 0,2 valores. Respostas em branco valem 0 valores.

Nas alíneas em que apenas uma resposta está correcta (se existirem estão assinaladas no texto), responder com mais ou menos do que um V anula a cotação. A resposta correcta corresponde à cotação completa. Respostas incorrectas correspondem a um desconto de metade da cotação completa.

Considere o seguinte programa:

```
#include <iostream>

using namespace std;

enum Género {masculino, feminino};

class Pessoa {
  public:
    Pessoa(int const idade = 18);
    virtual ~Pessoa();

    virtual Género género() const = 0;

    int idade() const;

  private:
    int idade_;
};

Pessoa::Pessoa(int const idade)
    : idade_(idade) {
}

virtual Pessoa::~Pessoa() {
}

int Pessoa::idade() const {
    return idade_;
}

class Homem : public Pessoa {
  public:

    Homem(int const idade = 18);
    virtual ~Homem();

    virtual Género género() const;
};

Homem::Homem(int const idade)
    : Pessoa(idade) {
}

Homem::~Homem() {
}

Género Homem::género() const {
    return masculino;
}

class Mulher : public Pessoa {
  public:

    Mulher(int const idade = 18);
    virtual ~ Mulher();

    virtual Género género() const;
};

Mulher::Mulher(int const idade)
    : Pessoa(idade) {
}

Mulher::~Mulher() {
}
```

Identificação
Nome do aluno: .....
Número do aluno: .....
Assinatura do docente: .....

```
Género Mulher::género() const {
    return feminino;
}
class Hermafrodita : public Homem, public Mulher {
  public:
    Hermafrodita(int const idade);
};
Hermafrodita::Hermafrodita(int const idade)
    : Homem(idade), Mulher(idade) {
}
…
template<typename T>
T const médiaDe(T const& x, T const& y) {
    return T((x + y) / 2.0);
}
…
int main()
{
    int xi = 2;
    int yi = 3;
    float xf = 3.0f;
    float yf = 4.0f;
    …
}
```

1.1 Quais das seguintes instruções estão correctas, admitindo que fazem parte da função `main()`, no local indicado pelas reticências?

- ☐ `int z = médiaDe(xi, yi);`
- ☐ `cout << "Média: " << médiaDe(xi, 5.3) << endl;`
- ☐ `cout << "Média: " << médiaDe<float>(2.0f, 3.0f) << endl;`
- ☐ `cout << "Média: " << médiaDe(2f, 3f) << endl;`
- ☐ `float zf = mediaDe(xi, yi);`

[cotação: 2,5]

1.2 Que surge na consola após a execução da seguinte instrução? Apenas uma resposta está correcta.

```
std::cout << médiaDe<int>(xf, yf)<<std::endl;
```

- ☐ `3`
- ☐ `3.5`
- ☐ `4`

[cotação: 0,5]

1.3 Quais das seguintes sequências de instruções estão correctas, admitindo que fazem parte da função `main()`, no local indicado pelas reticências?

- ☐
```
Pessoa zé(23);
```
- ☐
```
Homem manel;
```
- ☐
```
Hermafrodita maria_rapaz(10);
Género género = maria_rapaz.género();
```
- ☐
```
Hermafrodita maria_rapaz(10);
Género género = maria_rapaz.Homem::género();
```

[cotação: 2]

Identificação
Nome do aluno: .....
Número do aluno: .....
Assinatura do docente: .....

Questão 2

Pretende-se desenvolver um programa para gerir provas de atletismo, na modalidade de corrida.

Deve ser considerado o conceito de atleta, caracterizado pelo seu nome e altura. Devem existir operações para aceder a estas características. Cada atleta é obrigatoriamente considerado como sendo praticante de uma única modalidade: existirão atletas que são corredores, outros serão saltadores em comprimento, outros ainda serão lançadores do dardo. Os corredores são também caracterizados pela sua distância de especialização, e.g., 100 metros, 3000 metros barreiras ou maratona..

Um atleta pode pertencer ou não a uma equipa. Um atleta não pode pertencer a mais do que uma equipa. As equipas devem ser caracterizadas pelo seu nome e devem ter operações para contratar um novo atleta para a equipa ou dispensar um já existente, bem como para mostrar todos os atletas pertencentes à equipa. Deve também ser possível aceder a um atleta da equipa a partir do seu nome. A cada equipa pertencem apenas atletas de uma única modalidade. Existe sempre pelo menos um atleta numa equipa.

As provas de corrida devem aceitar inscrições de corredores, podendo cada corredor estar inscrito uma única vez. Devem igualmente permitir registar o tempo (em segundos) de cada um dos corredores inscritos, não podendo ser introduzido mais de um tempo por corredor. Deverão existir operações para mostrar os tempos gerais, bem como para aceder ao tempo de um corredor em particular, identificado pelo seu nome. É também útil poder saber se um dado corredor já está inscrito na corrida.

Leia o enunciado completo desta questão antes de começar a responder a cada uma das suas alíneas!

2.1 Desenhe o diagrama UML que descreve as relações entre as classes envolvidas neste programa. Pormenores que não sejam satisfatoriamente representados pelo diagrama, tais como restrições às associações entre classes, devem ser incluídos enquanto comentários. Nas classes, inclua as operações mas omita os atributos. Não inclua tão pouco pré-condições, condições objectivo e condições invariantes de classe.

[cotação: 3]

2.2 Defina as classes C++ necessárias à implementação do programa descrito. Não se esqueça de indicar a condição invariante de cada classe, bem como as pré-condições e condições objectivo de cada operação. As classes relativas a atletas e equipas devem encontrar-se definidas dentro de um espaço nominativo próprio.

Não implemente nenhum método!

[cotação: 4]

2.3 Implemente o método que testa a condição invariante de classe para a classe que representa uma equipa.

[cotação: 1]

2.4 Defina um construtor adequado para a classe que representa as equipas.
[cotação: 2]

2.5 Defina a operação que permite registar o tempo alcançado por um dos corredores.
[cotação: 2]

Identificação
Nome do aluno: .....
Número do aluno: .....
Assinatura do docente: .....

Questão 3

3.1 Durante a execução de um programa, podem ocorrer situações anormais ou imprevistas. Que tipos de situação anormais ou imprevistas conhece? Como se deve lidar com cada uma delas? Ilustre a sua resposta com exemplos concretos, nomeadamente com excertos de código.

[cotação: 3]