Recibo do Exame de 2ª Época de Introdução à Programação (IGE e ETI), 2003/07/23 **1º semestre de 2002/2003, ISCTE**
Nome do aluno: .....
Número do aluno: .....
Assinatura do docente: .....

Exame de 2ª Época

Introdução à Programação

IGE e ETI

2003/07/23 - 1º semestre de 2002/2003

ISCTE

Notas:

- Leia atentamente todo o enunciado (incluindo estas notas) antes de começar a prova.
- Este exame tem quatro 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 e uma caneta (coloque pastas, telemóveis e 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 18:00h de segunda-feira, dia 28 de Julho de 2003.
- A revisão de provas terá lugar das 14:30h às 15:30h de terça-feira, dia 29 de Julho de 2003, no gabinete D6.21.
- As orais terão lugar a partir das 8:00h de quinta-feira, dia 31 de Julho de 2003, no gabinete D6.18, devendo os interessados inscrever-se enviando uma mensagem de correio electrónico para Manuel.Sequeira@iscte.pt até às 12:00h de quarta-feira, dia 30 de Julho de 2003.
- 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 as declarações e o programa seguintes:

```
#include <cassert>
#include <iostream>
#include <vector>

using namespace std;

class Armario {
  public:
    Armario(int const numero_de_portas = 2);

    int numeroDePortas() const;
    bool estaAberto() const;
    bool estaAbertaAPorta(char const porta) const;

    void abrePorta(char const porta);
    void fechaPorta(char const porta);

    void abreAsPortasQueEstaoAbertasEm(Armario& armario);

    void mostraPortasAbertas() const;

  private:
    struct Porta {
        Porta(char const letra, bool const esta_aberta);
        char letra;
        bool esta_aberta;
    };

    vector<Porta> portas;

};


…


int main()
{
    Armario a(4);
    Armario const b(6);

    …

}
```

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

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

☐ `Armario c;`

☐ `Armario c('A', true);`

☐ `Armario c(b.numeroDePortas());`

[cotação: 1,5]

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

☐ `a = b;`

☐ `int const numero_de_portas = a.numeroDePortas();`

☐ `b.abrePorta('A');`

[cotação: 1,5]

1.3 Sabendo que `bool Armario::estaAberto() const` devolve verdadeiro quando todas as portas estão abertas e falso quando pelo menos uma delas está fechada, quais das seguintes definições estão correctas:

☐
```
bool Armario::estaAberto() const
{
    for(vector<Porta>::size_type i = 0; i != portas.size(); ++i)
        if(not portas[i].esta_aberta)
            return false;
        else
            return true;
}
```

☐
```
bool Armario::estaAberto() const
{
    vector<Porta>::size_type i = 0;
    while(i != portas.size() and portas[i].esta_aberta)
        ++i;
    return i == portas.size();
}
```

[cotação: 1]

1.4 Suponha as seguintes definições dos métodos acima declarados:

```
Armario::Armario(int const numero_de_portas) {
    for(int i = 0; i != numero_de_portas; ++i)
        portas.push_back(Porta(char(int('A') + i), false));
}

int Armario::numeroDePortas() const {
    return int(portas.size());
}

bool Armario::estaAbertaAPorta(char const porta) const {
    return portas[int(porta) - int('A')].esta_aberta;
}

void Armario::abrePorta(char const porta) {
    portas[int(porta) - int('A'))].esta_aberta = true;
}

void Armario::fechaPorta(char const porta) {
    portas[int(porta) - int('A')].esta_aberta = false;
}

void Armario::abreAsPortasQueEstaoAbertasEm(Armario& armario) {
    int numero_de_portas;
    if(numeroDePortas() < armario.numeroDePortas())
        numero_de_portas = numeroDePortas();
    else
        numero_de_portas = armario.numeroDePortas();
    for(int i = 0; i != numero_de_portas; ++i)
        if(armario.estaAbertaAPorta(char(int('A') + i))) {
            armario.fechaPorta(char(int('A') + i));
            abrePorta(char(int('A') + i));
        }
}

void Armario::mostraPortasAbertas() const {
    for(int i = 0; i != numeroDePortas(); ++i)
        if(portas[i].esta_aberta)
            cout << portas[i].letra << ' ';
    cout << endl;
}
```

Suponha o seguinte programa:

```
int main()
{
    Armario a(3), b(4);

    a.abrePorta('B');
    b.abrePorta('A');
    b.abrePorta('C');

    a.abreAsPortasQueEstaoAbertasEm(b);

    a.mostraPortasAbertas();
    b.mostraPortasAbertas();
}
```

Qual é o resultado da invocação do programa? *Apenas uma das opções está correcta.*

☐
```
B
A C
```

☐
```
A B C
```

☐
```
A B C
A C
```

☐
```
B
```

[cotação: 1]

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

Questão 2

Após o leccionar de uma matéria é necessário avaliar o conhecimento adquirido por cada aluno. Uma das ferramentas usadas no processo de avaliação é a prova com questões de escolha múltipla. A realização de uma prova deste tipo consiste na exibição progressiva de um enunciado, constituído por um dado número de questões de escolha múltipla. As respostas dadas ao longo da prova são guardadas e, no fim, o resultado obtido é apresentado ao aluno que realizou a prova.

O enunciado tem um determinado grau de dificuldade e é construído a partir de um repositório de questões. Cada questão tem um determinado texto, um conjunto de alternativas de resposta (das quais apenas uma está correcta), exibidos aquando da realização da prova, um dado grau de dificuldade e uma alternativa correcta. Os graus de dificuldade são apenas três: fácil, médio e difícil. Um enunciado de determinado grau de dificuldade deve ter 65% das questões com esse grau de dificuldade (o número de questões com esse grau de dificuldade deve ser o inteiro mais próximo dos 65% do número total das questões do enunciado). As questões de um enunciado são escolhidas aleatóriamente, respeitando o grau de dificuldade definido para o enunciado. Não podem haver questões com texto repetido.

As respostas dadas pelos alunos são números inteiros positivos, menores ou iguais ao número de alternativas da questão a que se dirige a resposta. Quando o aluno não quer responder, deve introduzir o valor 0 (zero).

O repositório de questões contém todas as perguntas disponíveis para a construção de enunciados. Não podem haver questões com texto repetido.

Pretende-se nesta questão construir e testar uma parte do sistema de realização de provas.

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

2.1 Defina a classe `Questão` que representa uma questão de escolha múltipla. Esta classe deve possuir as seguintes operações (operando sobre as suas instâncias):

- Construtor. Recebe o texto, as alternativas, a indicação de qual, das anteriores, é a alternativa correcta e o grau de dificuldade.
- Inspector para o texto da questão.
- Inspector para o número de alternativas.
- Predicado que dada uma resposta, devolve verdadeiro caso a resposta esteja correcta e falso no caso contrário. Uma resposta é um inteiro positivo, menor ou igual ao número de alternativas.
- Inspector para o grau de dificuldade.
- Inspector que mostre no ecrã a questão e as alternativas de resposta..

Indique qual é a condição invariante da classe e quais são as pré-condições e as condições objectivo das suas operações. Adicione a operação necessária para verificar a *CIC*.

Indique claramente quais das operações da classe não alteram a instância implícita.

Se necessitar de novos tipos de dados, estes também devem definidos.

Não é necessário nesta questão definir qualquer um dos métodos da classe.

[cotação: 1]

2.2 Defina a operação necessária para verificar a *CIC* da classe `Questão`.
[cotação: 1]

2.3 Defina completamente a classe `RepositórioDeQuestões`. Esta classe deve possuir as seguintes operações (operando sobre as suas instâncias):

- Modificador que adiciona uma questão. Não pode existir no repositório nenhuma questão com o mesmo texto.
- Inspector para o número de questões.
- Inspector para o número de questões com dado grau de dificuldade.
- Predicado que verifica se existe uma questão com o texto dado.
- Inspector que devolve uma questão aleatória.
- Inspector que devolve uma questão aleatória com dado grau de dificuldade.

Para obter uma questão aleatória, pode ser usada a função pré-existente `int rand()`. Esta função gera um número inteiro aleatório entre 0 e `RAND_MAX` (no caso da máquina em questão assume-se que é 2 147 483 647).

Indique qual é a condição invariante da classe e quais são as pré-condições e as condições objectivo das suas operações. Adicione a operação necessária para verificar a *CIC*.

Indique claramente quais das operações da classe não alteram a instância implícita.

Todas as condições (pré-condições, condições objectivo e condições invariantes de classe) devem ser verificadas através de instruções de asserção (`assert`). Nos métodos que mostram informação no ecrã não é necessário prestar atenção à formatação da saída.

[cotação: 2]

2.4 Defina completamente a classe `Enunciado`. Esta classe deve possuir as seguintes operações (operando sobre as suas instâncias):

- Construtor. Recebe um repositório de questões, o grau de dificuldade do enunciado e o número de questões do enunciado. O repositório deve conter questões suficientes para construir o enunciado e questões suficientes com o grau de dificuldade do exame (isto é, 65% do número de questões do enunciado). O número de questões do enunciado é um inteiro positivo.
- Inspector para o número de questões.
- Inspector para o número de questões com dado grau de dificuldade.
- Predicado que verifica se existe uma questão com o texto dado.
- Inspector que devolve o número de alternativas de resposta, dado o número de uma questão. O número da questão é um inteiro positivo, menor ou igual ao número de questões do enunciado.
- Predicado que, dada uma resposta e o número de uma questão, devolve verdadeiro caso a resposta seja a resposta correcta da questão dada e falso caso contrário. A resposta é um inteiro positivo, menor ou igual ao número de alternativas de resposta para essa questão e o número da questão é um inteiro positivo, menor ou igual ao número de questões do enunciado.
- Inspector que mostra uma questão, dado o número da questão. O número da questão é um inteiro positivo, menor ou igual ao número de questões do enunciado.

Indique qual é a condição invariante da classe e quais são as pré-condições e as condições objectivo das suas operações. Adicione a operação necessária para verificar a *CIC*.

Indique claramente quais das operações da classe não alteram a instância implícita.

Todas as condições (pré-condições, condições objectivo e condições invariantes de classe) devem ser verificadas através de instruções de asserção (`assert`). Nos métodos que mostram informação no ecrã não é necessário prestar atenção à formatação da saída.

[cotação: 3]

2.5 Defina a classe `Prova` que representa uma prova de questões de escolha múltipla. Esta classe deve possuir as seguintes operações (operando sobre as suas instâncias):

- Construtor. Recebe o enunciado.
- Inspector para o número de questões.
- Inspector para o número de respostas correctas.
- Inspector para o número de respostas erradas.
- Inspector para o número de questões por responder.
- Modificador que realiza a prova. A realização da prova consiste na exibição das perguntas do enunciado e na leitura das respostas dadas pelo aluno que realiza a prova, de uma forma iterativa.

Indique qual é a condição invariante da classe e quais são as pré-condições e as condições objectivo das suas operações. Adicione a operação necessária para verificar a *CIC*.

Indique claramente quais das operações da classe não alteram a instância implícita.

Não é necessário nesta questão definir qualquer um dos métodos da classe.

[cotação: 1]

2.6 Escreva um programa que permite simular a realização de uma prova.

O programa deve mostrar ao utilizador um menu que permita realizar as seguintes operações:

- Introduzir questões no repositório.
- Simular a realização de uma prova.

No caso de ser escolhida a primeira opção, o programa deve pedir os dados da questão e adicioná-la ao repositório.

Caso o utilizador escolha a segunda opção, o programa deve começar por perguntar o número de questões da prova e o nível de dificuldade. Em seguida, deve ser realizada a prova, e, por fim, exibidos os resultados: número de questões correctas, erradas e por responder.

De modo a garantir o bom funcionamento do programa, devem validar-se as entradas de dados. O programa deve garantir que nenhuma pré-condição é violada.

Não é necessário validar os tipos dos valores: se for pedido um valor inteiro, assume-se que o utilizador introduz um valor inteiro.

Para inicilizar a semente de geração de número aleatórios, a primeira instruçao do programa deve ser `srand(time(0))` (pertence a `cstdlib`).

Não se esqueça das directivas `#include` necessárias.

[cotação: 1]

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

Questão 3

Considere a função `int índiceDoPrimeiroMáximoDe(vector<int> const& v)` que devolve o índice do primeiro máximo do vector de inteiros `v`.

Desenvolva, utilizando um ciclo, uma implementação da função `índiceDoPrimeiroMáximoDe()`. Siga os passos da metodologia de Dijkstra pedidos nas alíneas desta questão.

Os seguintes cabeçalho e esqueleto do corpo da rotina devem ser respeitados:

```
int índiceDoPrimeiroMáximoDe(vector<int> const& v)
{
    int índice_do_primeiro_máximo = ...;
    // Ciclo:
    ...
    return índice_do_primeiro_máximo;
}
```

3.1 Indique a pré-condição (*PC*) e a condição objectivo (*CO*) da rotina e ainda a condição invariante (*CI*) do seu ciclo interno.

[cotação: 1]

3.2 Indique uma guarda *G* tal que *CI* e ¬*G* implica *CO*. Prove explicitamente que de facto *CI* e ¬*G* implica *CO*.

[cotação: 1]

3.3 Defina uma inicialização *inic* tal que se a *PC* for verdadeira, então a *CI* também é verdadeira após *inic* (inicialização das variáveis) e antes da primeira iteração do ciclo, i.e.,

$\{\!|\ PC\ |\!\}$

inic

$\{\!|\ \text{implica } CI\ |\!\}$.

Prove explicitamente que de facto:

$\{\!|\ PC\ |\!\}$

inic

$\{\!|\ \text{implica } CI\ |\!\}$.

[cotação: 1]

3.4 Defina um passo *passo* tal que se a guarda G e a CI do ciclo forem verdadeiras no início de um passo, então a CI também é verdadeira no fim desse passo, i.e.,

// *G* e *CI*

passo

// implica *CI*.

Prove explicitamente que de facto:

// *G* e *CI*

passo

// implica *CI*.

[cotação: 1]

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

Questão 4

4.1 Um dos conceitos mais importantes em programação é o de algoritmo. Explique o que é um algoritmo, indicando e descreva brevemente quais são as características dos algoritmos.

[cotação: 1]

4.2 O que é o contrato de uma rotina? Explicite sucintamente este conceito, ilustrando os papéis que um programador pode tomar no desenvolvimento de um programa.

[cotação: 1]