Este miniteste é composto por 4 questões em 4 páginas.
Responda em folhas **separadas**.

(5) 1. Considere o seguinte programa de C++:

```
1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  class Erro {};
6
7  template <class E> class RelBin {
8    vector<E> dominio, imagem;
9
10   int loc_par (const E & elem, const E & imag) const {
11       for (int i = 0; i < dominio.size(); i++)
12           if (dominio[i] == elem && imagem[i] == imag) return i;
13       return -1;
14   }
15
16 public:
17   RelBin &acrescenta_par(const E &elem, const E &imag) const {
18       if (loc_par(elem, imag) >= 0) throw Erro();
19         dominio.push_back(elem); imagem.push_back(imag);
20       return this;
21   }
22   void imprime_imagem(ostream &out, const E &elem) const {
23       for (int i = 0; i < dominio.size(); i++)
24           if (dominio[i] == elem) { imprime_par(out, i); out << '\n'; }
25       return 0;
26   }
27 };
28
29 int main()
30 {
31   RelBin<int> rel;
32   try {
33         rel.acrescenta_par(1,3).acrescenta_par(2,9).acrescenta_par(2,7);
34         rel.acrescenta_par(3,2).acrescenta_par(2,9).acrescenta_par(2,5);
35   }
36   catch (Erro) { cout << "XX!!\n"; }
37   rel.imprime_imagem(cout, 2);
38   return 0;
39 }
```

(a) O programa apresentado contém três erros sintácticos. Detecte-os e corrija-os.

(b) Indique o que surge no monitor quando o programa corrigido é executado.

(6) 2. Pretende-se implementar uma classe que representa temperaturas. Cada temperatura pode ser dada em graus Kelvin, Celsius ou Fahrenheit. Internamente, a temperatura é representada **sempre** em graus Kelvin. Considere as seguintes declarações:

```
class Erro {};
class Temperatura {
  double kelvin;   // a temperatura SEMPRE em graus Kelvin
  static double c_para_k (double c) { /* falta */ }
  static double f_para_k (double f) { /* falta */ }
  static double k_para_c (double k) { /* falta */ }
  static double k_para_f (double k) { /* falta */ }
  static double para_k(double valarg, char unidade);
  static double de_k(double valarg, char unidade);
public:
  Temperatura(double val = 0.0, char unidade = 'C');
  double operator[](const char unidade) const;
  bool operator==(const Temperatura &t) const;
  Temperatura & somar(double var, char unidade = 'C');
  void escrever(ostream &os, const char unidade = 'C') const;
};
ostream &operator<<(ostream &os, const Temperatura &t);
```

(a) Escreva os quatro membros-função auxiliares indicados na tabela seguinte.

`c_para_k()`	converte de graus Celsius para graus Kelvin
`f_para_k()`	converte de graus Fahrenheit para graus Kelvin
`k_para_c()`	converte de graus Kelvin para graus Celsius
`k_para_f()`	converte de graus Kelvin para graus Fahrenheit

As relações entre as diferentes unidades são as seguintes:

$$K = (F + 459.67)/1.8$$

$$K = C + 273.15$$

(b) Usando as funções da alínea anterior, defina os membros-função auxiliares `para_k()` e `de_k()`. O membro-funcção `para_k()` converte o seu argumento para graus Kelvin. O membro-função `de_k()` converte o valor do primeiro argumento (que é suposto estar expresso em graus Kelvin) para a unidade desejada (o segundo argumento). As unidades são indicadas por um carácter: `'K'` para Kelvin, `'F'` para Fahrenheit e `'C'` para Celsius. Caso a unidade seja desconhecida, estas funções devem lançar uma excepção do tipo `Erro`.

(c) Implemente o construtor.

(d) Implemente o operador de igualdade.

(e) Implemente o membro-função `operator[]` que retorna o valor da temperatura nas unidades indicas como argumento.

(f) Implemente o membro-função `somar()` que adiciona à temperatura a quantidade especificada em argumento (tendo em atenção a unidade em que a variação é expressa).

(g) Implemente o membro-função `escrever()` que envia para um fluxo de saída o valor da temperatura seguido de um carácter a indicar a unidade. O valor da temperatura deve ser convertido para corresponder à unidade pedida.

(h) Implemente a função `operator<<()`. A temperatura é apresentada em graus Celsius.

(5) 3. Considere o seguinte programa em C++:

```
#include <iostream>
using namespace std;

class Medida {
protected:
  double valor;
public:
  Medida(double n): valor(n) {};
  virtual double val() { return valor; }
  double simetrico() { return -valor; }
};

class MedidaEscala: public Medida {
protected:
  double escala;
public:
  MedidaEscala(double n, double m = 2.0): Medida(n), escala(m) {};
  virtual double val() { return valor*escala; }
  double simetrico() { return -(valor*escala); }
};

class MedidaDeslocamento: public Medida {
protected:
  double deslocamento;
public:
  MedidaDeslocamento(double n, double d = 1.0): Medida(n), deslocamento(d) {};
  virtual double val() { return valor+deslocamento; }
  double simetrico() { return -(valor+deslocamento); }
};

double g(Medida *obj) { return obj->val(); }
double f(Medida *obj) { return obj->simetrico(); }
double h(Medida obj) { return obj.val(); }

int main()
{
  Medida a(10.0); MedidaEscala  b(10.0);  MedidaDeslocamento  c(10.0);
  cout << a.val() << ' ' << a.simetrico() << ' ' << f(&a)
       << ' ' << g(&a) << ' ' << h(a) << endl;
  cout << b.val() << ' ' << b.simetrico() << ' ' << f(&b)
       << ' ' << g(&b) << ' ' << h(b) << endl;
  cout << c.val() << ' ' << c.simetrico() << ' ' << f(&c)
       << ' ' << g(&c) << ' ' << h(c) << endl;
  return 0;
}
```

Indique o que aparece no monitor quando o programa é executado. Justifique.

(4) 4. Pretende-se avaliar a complexidade temporal e espacial da seguinte função genérica escrita em C++, que determina se os elementos de um vector estão contidos noutro vector em sequência contígua. Assuma que o primeiro vector tem N elementos e o segundo M elementos, com N>>M.

```
template <class T>
bool contido(const vector<T> & v1, const vector<T> & v2)
{
  int niter =  v1.size()-v2.size();
  int i, j;
  for (i=0; i <= niter; i++) {
        for (j=0; j < v2.size(); j++) {
          if (v1[i+j] != v2[j])
                break;
        }
        if (j==v2.size()) return true;
  }
  return false;
}
```

(a) Assumindo que T representa um tipo básico (`int`, `double`, etc.), determine a complexidade temporal e espacial da função. Justifique.

(b) Como deve ser generalizado o resultado da alínea anterior para um tipo T qualquer? Justifique.

Fim.