Início	quarta, 15 de junho de 2022 às 09:17
Estado	Prova submetida
Data de submissão:	quarta, 15 de junho de 2022 às 10:36
Tempo gasto	1 hora 18 minutos
Nota	**100,0** do máximo 100,0

Pergunta 1

Correta Pontuou 20,0 de 20,0

Write a C++ function `void maximum(const string& input_fname, const string& output_fname)` that reads `double` values with variable number of decimal places, stored one per line in input file named `input_fname`, and outputs to file name `output_fname` the corresponding values, one per line, rounded to 3 decimal places. In the end, the function outputs the *number of values* read and the *maximum value* (see examples in the tests). You may assume that the values are all in the range [-1000,1000].

To test your code download the ex1.zip archive containing the text files used in public tests (`ex1-[1-4].txt`). You may assume that the only blank characters contained in files are the space and newline character.

Por exemplo:

Teste	Resultado

`maximum("ex1-1.txt", "ex1-1_out.txt");`
`show_file("ex1-1_out.txt");`
	`==> ex1-1_out.txt <==`
	`-1.200`
	`1.222`
	`3.142`
	`count=3/max=3.142`

`maximum("ex1-2.txt", "ex1-2_out.txt");`
`show_file("ex1-2_out.txt");`
	`==> ex1-2_out.txt <==`
	`-11.223`
	`-65.240`
	`-3.142`
	`count=3/max=-3.142`

`maximum("ex1-3.txt", "ex1-3_out.txt");`
`show_file("ex1-3_out.txt");`
	`==> ex1-3_out.txt <==`
	`-11.000`
	`0.000`
	`12.452`
	`123.457`
	`count=4/max=123.457`

`maximum("ex1-4.txt", "ex1-4_out.txt");`
`show_file("ex1-4_out.txt");`
	`==> ex1-4_out.txt <==`
	`1.223`
	`1.200`
	`3.146`
	`count=3/max=3.146`

Resposta: (regime de penalização: 0, 0, 0, 0, 10, 20, 30, ... %)

Limpar resposta

```cpp
#include <iostream>
#include <iomanip>
#include <fstream>
#include <string>
#include <sstream>

using namespace std;

//! Show file name and its contents.
void show_file(const string& file) {
  ifstream in(file);
  cout << "==> " << file << " <==\n";
  for (string line; getline(in, line); ) cout << line << '\n';
}

void maximum(const string& input_fname, const string& output_fname){
    ifstream reader(input_fname);
    ofstream writter(output_fname);
    string line;
    int count = 0;
    double max = -1000;
    while(getline(reader, line)){
```

	Teste	Esperado	Recebido	

	Teste	Esperado	Recebido	

✓	`maximum("ex1-1.txt", "ex1-1_out.txt");`
	`show_file("ex1-1_out.txt");`
		`==> ex1-1_out.txt <==`
		`-1.200`
		`1.222`
		`3.142`
		`count=3/max=3.142`
			`==> ex1-1_out.txt <==`
			`-1.200`
			`1.222`
			`3.142`
			`count=3/max=3.142`
				✓

✓	`maximum("ex1-2.txt", "ex1-2_out.txt");`
	`show_file("ex1-2_out.txt");`
		`==> ex1-2_out.txt <==`
		`-11.223`
		`-65.240`
		`-3.142`
		`count=3/max=-3.142`
			`==> ex1-2_out.txt <==`
			`-11.223`
			`-65.240`
			`-3.142`
			`count=3/max=-3.142`
				✓

✓	`maximum("ex1-3.txt", "ex1-3_out.txt");`
	`show_file("ex1-3_out.txt");`
		`==> ex1-3_out.txt <==`
		`-11.000`
		`0.000`
		`12.452`
		`123.457`
		`count=4/max=123.457`
			`==> ex1-3_out.txt <==`
			`-11.000`
			`0.000`
			`12.452`
			`123.457`
			`count=4/max=123.457`
				✓

✓	`maximum("ex1-4.txt", "ex1-4_out.txt");`
	`show_file("ex1-4_out.txt");`
		`==> ex1-4_out.txt <==`
		`1.223`
		`1.200`
		`3.146`
		`count=3/max=3.146`
			`==> ex1-4_out.txt <==`
			`1.223`
			`1.200`
			`3.146`
			`count=3/max=3.146`
				✓

Passou em todos os testes! ✓

Solução do autor da pergunta (C):

```cpp
#include <iostream>
#include <iomanip>
#include <fstream>
#include <cfloat>

using namespace std;

//! Show file name and its contents.
void show_file(const string& file) {
  ifstream in(file);
  cout << "==> " << file << " <==\n";
  for (string line; getline(in, line); ) cout << line << '\n';
}
// <-- Answer preload

void maximum(const string& input_fname, const string& output_fname) {
  ifstream f_in(input_fname);
  ofstream f_out(output_fname);
  double num;
  int count = 0;
  double max_num = -1000;
  while (f_in >> num) {
```

Correta

Nota desta submissão: 20,0/20,0

Pergunta 2

Correta Pontuou 20,0 de 20,0

Write the C++ code for the `Student` class that represents a student in the Bachelor in Informatics and Computing Engineering, with the definition given in the `Student_b.h` header file.

```
struct course {
  std::string name;  // "ALGA", "AMI", "FSC", "MD", ...
  float credits;     // 1.5, 4.5, 6
  short grade;       // 0..20
};

class Student {
public:
  // constructor with parameters
  Student(const std::string& id);
  // accessor
  std::string get_id() const;
  // member functions
  void add(const course& c);  // add a course grade
  double avg() const;         // calculate the average grade
private:
  std::string id_;
  std::vector<course> courses_;
};
```

Note that the `avg` function must return the *weighted average grade* calculated by multiplying the grades by its corresponding credits and taking its sum:

$$\frac{\sum_{i=1}^{n} credit_i * grade_i}{\sum_{i=1}^{n} credit_i}$$

You can assume that only courses with passing grades (`grade>=10`) are added.

Por exemplo:

Teste	Resultado

`Student s("up2022001111");`
`s.add({"FP", 6, 20});`
`cout << "[" << s.get_id() << "]/";`
`cout << fixed << setprecision(2) << s.avg() << endl;`
	`[up2022001111]/20.00`

`Student s("up2022002222");`
`s.add({"MD", 6, 18}); s.add({"AMI", 6, 16}); s.add({"FSC", 6, 14});`
`cout << "[" << s.get_id() << "]/";`
`cout << fixed << setprecision(2) << s.avg() << endl;`
	`[up2022002222]/16.00`

`Student s("up2022003333");`
`s.add({"ALGA", 4.5, 18}); s.add({"AMI", 6, 16}); s.add({"PUP", 1.5, 14});`
`cout << "[" << s.get_id() << "]/";`
`cout << fixed << setprecision(2) << s.avg() << endl;`
	`[up2022003333]/16.50`

`Student s("up2022004444");`
`cout << "[" << s.get_id() << "]/";`
`cout << fixed << setprecision(2) << s.avg() << endl;`
	`[up2022004444]/0.00`

Resposta: (regime de penalização: 0, 0, 0, 0, 10, 20, 30, ... %)

Limpar resposta

```
#include <iostream>
#include <iomanip>
#include "Student_b.h"

using namespace std;

Student::Student(const string& id){
    id_ = id;
}

string Student::get_id() const{
    return id_;
}

void Student::add(const course& c){
    courses_.push_back(c);
```

```
}

double Student::avg() const{
    double average = 0;
    double total_credits = 0;
    for(course c: courses_){
```

	Teste	Esperado	Recebido	

✓	`Student s("up2022001111");`
	`s.add({"FP", 6, 20});`
	`cout << "[" << s.get_id() << "]/";`
	`cout << fixed << setprecision(2) << s.avg() << endl;`
		[up2022001111]/20.00	[up2022001111]/20.00	✓

✓	`Student s("up2022002222");`
	`s.add({"MD", 6, 18}); s.add({"AMI", 6, 16}); s.add({"FSC", 6, 14});`
	`cout << "[" << s.get_id() << "]/";`
	`cout << fixed << setprecision(2) << s.avg() << endl;`
		[up2022002222]/16.00	[up2022002222]/16.00	✓

✓	`Student s("up2022003333");`
	`s.add({"ALGA", 4.5, 18}); s.add({"AMI", 6, 16});`
	`s.add({"PUP", 1.5, 14});`
	`cout << "[" << s.get_id() << "]/";`
	`cout << fixed << setprecision(2) << s.avg() << endl;`
		[up2022003333]/16.50	[up2022003333]/16.50	✓

✓	`Student s("up2022004444");`
	`cout << "[" << s.get_id() << "]/";`
	`cout << fixed << setprecision(2) << s.avg() << endl;`
		[up2022004444]/0.00	[up2022004444]/0.00	✓

Passou em todos os testes! ✓

Solução do autor da pergunta (C):

```
#include <iostream>
#include <iomanip>
#include "Student_b.h"

using namespace std;

Student::Student(const std::string& id) : id_(id) { }

std::string Student::get_id() const { return id_; }

void Student::add(const course& c) { courses_.push_back(c); }

double Student::avg() const {
  int sum = 0;
  float crd = 0;
  for (course c : courses_) {
    sum += c.credits * c.grade;
    crd += c.credits;
  }
  return (crd==0) ? 0 : (double) sum/crd;
}

```

Correta

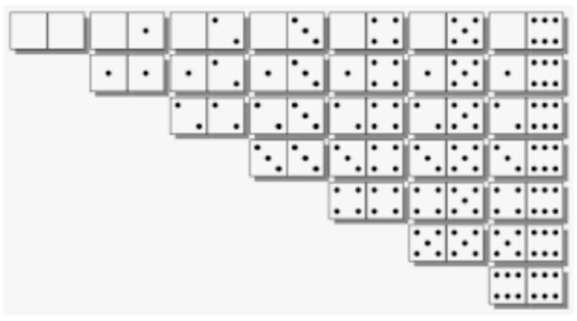
Nota desta submissão: 20,0/20,0

Pergunta 3

Correta

Pontuou 20,0 de 20,0

Dominoes is a well known game.



Each piece of a domino game is a rectangular **tile** with a line dividing its face into two sides marked with a number of spots between **0** and **6**.

The set of available tiles, to be collected by the players during the game, is called the **deck**, and contains up to 28 tiles corresponding to all possible combinations of spot counts between 0 and 6.

A partial definition of the classes `Tile` and `Deck` is given in Tile.h and Deck.h, respectively.

```
//! Represents a tile.
class Tile {
public:
  // Constructor
  Tile(int s1, int s2) : s1_(s1), s2_(s2) { }
  // Test if tile matches t in any of the sides
  bool compatible_with(const Tile& t) const; // TO BE IMPLEMENTED
  // Show tile
  void show() const { cout << s1_ << '-' << s2_; }
private:
  int s1_, s2_;
};

//! Represents available tiles.
class Deck {
public:
  // Constructor
  Deck(const vector<Tile>& tiles) : tiles_(tiles) {}
  // Remove all tiles compatible with given tile
  int remove_all_compatible_with(const Tile& t); // TO BE IMPLEMENTED
  // Show tiles in deck
  void show() const {
    cout << "[ ";
    for (auto t : tiles_) { t.show(); cout << ' '; }
    cout << "]\n";
  }
private:
  vector<Tile> tiles_;
};
```

Implement the two missing member functions, `Tile::compatible_with` and `Deck::remove_all_compatible_with`, such that:

- `t1.compatile_with(t2)` returns `true` if tiles `t1` and `t2` have at least one side in common — note that the order of the spots in terms of the `s1_` and `s2_` fields is not relevant; and
- `d.remove_all_compatible_with(t)` removes all tiles from deck `d` that are compatible with tile `t` and returns the number of removed tiles.

Por exemplo:

Teste	Resultado

Teste	Resultado

`Tile t(1, 2);`
`cout << boolalpha`
`     << t.compatible_with({1,2}) << ' '`
`     << t.compatible_with({3,4}) << ' '`
`     << t.compatible_with({3,1}) << ' '`
`     << t.compatible_with({1,3}) << ' '`
`     << t.compatible_with({2,3}) << ' '`
`     << t.compatible_with({3,2}) << ' '`
`     << t.compatible_with({0,5}) << ' '`
`     << t.compatible_with({6,6}) << '\n';`
	`true false true true true true false false`

`Tile t(0, 6);`
`Deck d({ { 1, 2}, {3, 4} });`
`cout << d.remove_all_compatible_with(t) << ' ';`
`d.show();`
	`0 [ 1-2 3-4 ]`

`Tile t(0, 6);`
`Deck d({ {0, 6}, {3, 6}, {5, 0}, {0, 0} });`
`cout << d.remove_all_compatible_with(t) << ' ';`
`d.show();`
	`4 [ ]`

`Tile t(1, 2);`
`Deck d({ {1, 2}, {3, 4}, {3, 2}, {1, 5}, {5, 6}, { 6, 6} });`
`cout << d.remove_all_compatible_with(t) << ' ';`
`d.show();`
	`3 [ 3-4 5-6 6-6 ]`

`Tile t(6, 6);`
`Deck d({ {1, 2}, {3, 4}, {3, 2}, {1, 5}, {5, 2}, { 6, 6} });`
`cout << d.remove_all_compatible_with(t) << ' ';`
`d.show();`
	`1 [ 1-2 3-4 3-2 1-5 5-2 ]`

`vector<Tile> v;`
`for (int i = 0; i <= 6; i++) for (int j = 6; j >= i; j--)`
`v.push_back({ j, i });`
`Deck d(v);`
`cout << d.remove_all_compatible_with({ 5, 5}) << ' '`
`     << d.remove_all_compatible_with({ 1, 2}) << ' ';`
`d.show();`
	`7 11 [ 6-0 4-0 3-0 0-0 6-3 4-3 3-3 6-4 4-4 6-6 ]`

Resposta: (regime de penalização: 0, 0, 0, 0, 10, 20, 30, ... %)

Limpar resposta

```cpp
#include <iostream>
#include <iomanip>
#include <vector>
#include "Tile.h"
#include "Deck.h"

bool Tile::compatible_with(const Tile& t) const{
    return(s1_ == t.s1_ || s1_ == t.s2_ || s2_ == t.s1_ || s2_ == t.s2_);
}

int Deck::remove_all_compatible_with(const Tile& t){
    vector<Tile> vnew;
    int result = 0;
    for(Tile t2 : tiles_){
        if(t2.compatible_with(t)){
            result++;
        }
        else{
            vnew.push_back(t2);
        }
    }
    tiles_ = vnew;
```

	Teste	Esperado	Recebido	

	Teste	Esperado	Recebido	

✓	`Tile t(1, 2);`
	`cout << boolalpha`
	`     << t.compatible_with({1,2}) << ' '`
	`     << t.compatible_with({3,4}) << ' '`
	`     << t.compatible_with({3,1}) << ' '`
	`     << t.compatible_with({1,3}) << ' '`
	`     << t.compatible_with({2,3}) << ' '`
	`     << t.compatible_with({3,2}) << ' '`
	`     << t.compatible_with({0,5}) << ' '`
	`     << t.compatible_with({6,6}) << '\n';`
		`true false true true true`
		`true false false`
			`true false true true true`
			`true false false`
				✓

✓	`Tile t(0, 6);`
	`Deck d({ { 1, 2}, {3, 4} });`
	`cout << d.remove_all_compatible_with(t) << '`
	`';`
	`d.show();`
		`0 [ 1-2 3-4 ]`	`0 [ 1-2 3-4 ]`	✓

✓	`Tile t(0, 6);`
	`Deck d({ {0, 6}, {3, 6}, {5, 0}, {0, 0} });`
	`cout << d.remove_all_compatible_with(t) << '`
	`';`
	`d.show();`
		`4 [ ]`	`4 [ ]`	✓

✓	`Tile t(1, 2);`
	`Deck d({ {1, 2}, {3, 4}, {3, 2}, {1, 5}, {5,`
	`6}, { 6, 6} });`
	`cout << d.remove_all_compatible_with(t) << '`
	`';`
	`d.show();`
		`3 [ 3-4 5-6 6-6 ]`	`3 [ 3-4 5-6 6-6 ]`	✓

✓	`Tile t(6, 6);`
	`Deck d({ {1, 2}, {3, 4}, {3, 2}, {1, 5}, {5,`
	`2}, { 6, 6} });`
	`cout << d.remove_all_compatible_with(t) << '`
	`';`
	`d.show();`
		`1 [ 1-2 3-4 3-2 1-5 5-2 ]`	`1 [ 1-2 3-4 3-2 1-5 5-2 ]`	✓

✓	`vector<Tile> v;`
	`for (int i = 0; i <= 6; i++) for (int j = 6;`
	`j >= i; j--) v.push_back({ j, i });`
	`Deck d(v);`
	`cout << d.remove_all_compatible_with({ 5, 5})`
	`<< ' '`
	`     << d.remove_all_compatible_with({ 1, 2})`
	`<< ' ';`
	`d.show();`
		`7 11 [ 6-0 4-0 3-0 0-0 6-3`
		`4-3 3-3 6-4 4-4 6-6 ]`
			`7 11 [ 6-0 4-0 3-0 0-0 6-3`
			`4-3 3-3 6-4 4-4 6-6 ]`
				✓

Passou em todos os testes! ✓

Solução do autor da pergunta (C):

```
#include <iostream>
#include <iomanip>
#include "Tile.h"
#include "Deck.h"

using namespace std;

//! determine if this piece can be placed on the left of other
bool Tile::compatible_with(const Tile& other) const {
  return s1_ == other.s1_ || s1_ == other.s2_ ||
         s2_ == other.s1_ || s2_ == other.s2_;
}

#if 0
// Solution that makes use of an iterator
int Deck::remove_all_compatible_with(const Tile& t) {
  int r = 0;
  for (auto itr = tiles_.begin(); itr != tiles_.end();) {
    if (t.compatible_with(*itr)) {
      r++;
      itr = tiles_.erase(itr);
    } else {
```

Correta

Nota desta submissão: 20,0/20,0

Pergunta 4

Correta Pontuou 20,0 de 20,0

Write the C++ code for function `smallest_sum_key`, declared as:

```
string smallest_sum_key(map<string, list<int>> m);
```

that returns the key that maps to the list with the *smallest sum*. You may consider that exactly one list has the smallest sum.

Note that `INT_MAX`, defined in header `<climits>`, is the constant for the maximum value of an `int` value.

Por exemplo:

Teste	Resultado

`map<string, list<int>> m1 = {`
`  {"s1", {1, 2, 3} } };`
`cout << smallest_sum_key(m1) << endl;`
	s1

`map<string, list<int>> m2 = {`
`  {"s1", {1, 2, 3} }, {"s2", {2, 3, 4} } };`
`cout << smallest_sum_key(m2) << endl;`
	s1

`map<string, list<int>> m3 = {`
`  {"s1", {10, 100} }, {"s2", {101, 100, 100} }, {"s3", {100, 5, 4} } };`
`cout << smallest_sum_key(m3) << endl;`
	s3

`map<string, list<int>> m4 = {`
`  {"s1", {-100, -100} }, {"s2", {-200, -100} } };`
`cout << smallest_sum_key(m4) << endl;`
	s2

Resposta: (regime de penalização: 0, 0, 0, 0, 10, 20, 30, ... %)

Limpar resposta

```
#include <map>
#include <list>
#include <string>
#include <iostream>
#include <climits>

using namespace std;

string smallest_sum_key(map<string, list<int>> m){
    string result;
    int min = INT_MAX;
    for(auto it = m.begin(); it != m.end(); it++){
        list<int> l = it->second;
        int sum = 0;
        for(auto itl = l.begin(); itl !=l.end(); itl++){
            sum += *itl;
        }
        if(sum < min){
            min = sum;
            result = it->first;
        }
    }
```

	Teste	Esperado	Recebido	

✓	`map<string, list<int>> m1 = {`
	`  {"s1", {1, 2, 3} } };`
	`cout << smallest_sum_key(m1) << endl;`
		s1	s1	✓

✓	`map<string, list<int>> m2 = {`
	`  {"s1", {1, 2, 3} }, {"s2", {2, 3, 4} } };`
	`cout << smallest_sum_key(m2) << endl;`
		s1	s1	✓

✓	`map<string, list<int>> m3 = {`
	`  {"s1", {10, 100} }, {"s2", {101, 100, 100} }, {"s3", {100, 5, 4} } };`
	`cout << smallest_sum_key(m3) << endl;`
		s3	s3	✓

✓	`map<string, list<int>> m4 = {`
	`  {"s1", {-100, -100} }, {"s2", {-200, -100} } };`
	`cout << smallest_sum_key(m4) << endl;`
		s2	s2	✓

Passou em todos os testes! ✓

Solução do autor da pergunta (C):

```
#include <map>
#include <list>
#include <string>
#include <iostream>
#include <climits>

using namespace std;

string smallest_sum_key(const map<string, list<int>> m) {
  string smallest_key;
  int smallest_sum = INT_MAX;
  for (auto s : m) {
    int this_sum = 0;
    for (int val : s.second)
      this_sum += val;
    if (this_sum < smallest_sum) {
      smallest_sum = this_sum;
      smallest_key = s.first;
    }
  }
  return smallest_key;
}
```

Correta

Nota desta submissão: 20,0/20,0

Pergunta 5

Correta Pontuou 20,0 de 20,0

Consider the definition of an abstract class `Account` given in header file `Account.h`, that represents bank accounts.

```
class Account {
  public:
    Account(int number) : number_(number) { }
    int get_number() const { return number_; }
    virtual float get_balance() const = 0;
  protected:
    int number_; // account number
};
```

Implement the definition of classes `Regular` and `Deposits`, derived from `Account`.

`Regular` represents bank accounts for daily usage; they should hold a floating point value, initially 0, for their balance and provide an implementation of:

```
//! adds amount to the balance
void increase_balance(float amount);
```

`Deposits` represents bank accounts for long term savings; they should hold a sequence of floating point values, initially empty, each with the amount of one deposit, and provide an implementation of:

```
//! adds a deposit to the list of deposits
void add_deposit(float amount);

//! accesses the deposit at index [deposit_number-1] in the list
float get_deposit(int deposit_number) const;
```

Por exemplo:

Teste	Resultado

`Regular a1(10001); const Account& r = a1;`
`float v = r.get_balance();`
`a1.increase_balance(34.50);`
`cout << fixed << setprecision(2)`
`     << r.get_number() << ' ' << v << ' ' << r.get_balance() << endl;`
	`10001 0.00 34.50`

`Regular a1(10002);`
`a1.increase_balance(34.50); a1.increase_balance(34.50);`
`cout << fixed << setprecision(2)<< a1.get_balance() << endl;`
	`69.00`

`Deposits a2(20001); const Account& r = a2; const Deposits& r2 = a2;`
`float v = r.get_balance();`
`a2.add_deposit(39.50); a2.add_deposit(12.30);`
`cout << fixed << setprecision(2)`
`     << r.get_number() << ' '`
`     << v << ' '`
`     << r.get_balance() << ' '`
`     << r2.get_deposit(1) << ' '`
`     << r2.get_deposit(2) << endl;`
	`20001 0.00 51.80 39.50 12.30`

`Deposits a2(20002);`
`a2.add_deposit(1000.50); a2.add_deposit(100.50);`
`cout << fixed << setprecision(2)<< a2.get_balance() << ' ' << a2.get_deposit(2) << endl;`
	`1101.00 100.50`

`Regular a1(99); a1.increase_balance(1.2);`
`Deposits a2(100); a2.add_deposit(1.2); a2.add_deposit(3.4); a2.add_deposit(5.6);`
`const Account& ra1 = a1; const Account& ra2 = a2; const Deposits& ra2_b = a2;`
`cout << fixed << setprecision(3)`
`     << ra1.get_number() << ' ' << ra2.get_number() << ' '`
`     << ra1.get_balance() << ' ' << ra2.get_balance() << ' '`
`     << ra2_b.get_deposit(1) << ' ' << ra2_b.get_deposit(2) << ' ' << ra2_b.get_deposit(3) << '\n';`
	`99 100 1.200 10.200 1.200 3.400 5.600`

Resposta: (regime de penalização: 0, 0, 0, 0, 10, 20, 30, ... %)

Limpar resposta

```
// Answer Preload -->
#include <iostream>
#include <iomanip>
```

```
#include <vector>
#include "Account.h"

using namespace std;
// <-- Answer Preload

class Regular:public Account{
    public:
        Regular(int n) : Account(n){
            amount_ = 0;
        }
        void increase_balance(float amount){
            amount_ += amount;
        }
        float get_balance() const override{
            return amount_;
        }
    private:
        float amount_;
```

	Teste	Esperado	Recebido	

✓	`Regular a1(10001); const Account& r = a1;`
	`float v = r.get_balance();`
	`a1.increase_balance(34.50);`
	`cout << fixed << setprecision(2)`
	`     << r.get_number() << ' ' << v << ' ' <<`
	`r.get_balance() << endl;`
		`10001 0.00 34.50`	`10001 0.00 34.50`	✓

✓	`Regular a1(10002);`
	`a1.increase_balance(34.50); a1.increase_balance(34.50);`
	`cout << fixed << setprecision(2)<< a1.get_balance() <<`
	`endl;`
		`69.00`	`69.00`	✓

✓	`Deposits a2(20001); const Account& r = a2; const`
	`Deposits& r2 = a2;`
	`float v = r.get_balance();`
	`a2.add_deposit(39.50); a2.add_deposit(12.30);`
	`cout << fixed << setprecision(2)`
	`     << r.get_number() << ' '`
	`     << v << ' '`
	`     << r.get_balance() << ' '`
	`     << r2.get_deposit(1) << ' '`
	`     << r2.get_deposit(2) << endl;`
		`20001 0.00 51.80`
		`39.50 12.30`
			`20001 0.00 51.80`
			`39.50 12.30`
				✓

✓	`Deposits a2(20002);`
	`a2.add_deposit(1000.50); a2.add_deposit(100.50);`
	`cout << fixed << setprecision(2)<< a2.get_balance() << '`
	`' << a2.get_deposit(2) << endl;`
		`1101.00 100.50`	`1101.00 100.50`	✓

✓	`Regular a1(99); a1.increase_balance(1.2);`
	`Deposits a2(100); a2.add_deposit(1.2);`
	`a2.add_deposit(3.4); a2.add_deposit(5.6);`
	`const Account& ra1 = a1; const Account& ra2 = a2; const`
	`Deposits& ra2_b = a2;`
	`cout << fixed << setprecision(3)`
	`     << ra1.get_number() << ' ' << ra2.get_number() << '`
	`'`
	`     << ra1.get_balance() << ' ' << ra2.get_balance() <<`
	`' '`
	`     << ra2_b.get_deposit(1) << ' ' <<`
	`ra2_b.get_deposit(2) << ' ' << ra2_b.get_deposit(3) <<`
	`'\n';`
		`99 100 1.200 10.200`
		`1.200 3.400 5.600`
			`99 100 1.200 10.200`
			`1.200 3.400 5.600`
				✓

Passou em todos os testes! ✓

Solução do autor da pergunta (C):

```
#include <iostream>
#include <iomanip>
#include <vector>
#include "Account.h"

using namespace std;
```

```

class Regular : public Account {
  public:
    Regular(int number) : Account(number), r_balance_(0) { }
    //! adds amount to the balance
    void increase_balance(float amount) { r_balance_ += amount; }
    float get_balance() const override{ return r_balance_; }
  private:
    float r_balance_;
};

class Deposits : public Account {
  public:
    Deposits(int number): Account(number) { }
    //! adds a deposit to the list of deposits
    void add_deposit(float amount) {
```

Correta

Nota desta submissão: 20,0/20,0