Início	terça, 17 de maio de 2022 às 08:42
Estado	Prova submetida
Data de submissão:	quinta, 19 de maio de 2022 às 20:17
Tempo gasto	2 dias 11 horas
Nota	**100** do máximo 100

Pergunta 1

Correta Pontuou 20 de 20

Write a C++ class named `Color` to represent colors using the Red-Green-Blue (RGB) color model such that each RGB component is an 8-bit unsigned integer (unsigned char, values from 0 to 255).

You must **submit only two files as attachments**: `Color.h` and `Color.cpp`. The `Color.h` file should contain only the class declaration, and `Color.cpp` should contain the actual implementation.

In your local workspace, to compile a program in a file containing the tests, named for instance `main.cpp`, you should use the supplied `Makefile` and execute the following command in the Terminal:

```
make PROG=main CPP_FILES="Color.cpp main.cpp" HEADERS="Color.h"
```

The `Color` class should have:

- A constructor
  `Color(unsigned char red, unsigned char green, unsigned char blue);`
  that takes as argument the RGB values to use for the color
- A copy constructor
  `Color(const Color& c);`
- Accessors:
  `unsigned char red() const;`
  `unsigned char green() const;`
  `unsigned char blue() const;`
- Static class constants
  `static const Color RED, GREEN, BLUE, BLACK, WHITE;`
  corresponding to colors red (255-0-0 in RGB), green (0-255-0), blue (0-0-255), black (0-0-0), and white (255-255-255)
- a member function to test equality between colors
  `bool equal_to(const Color& other) const;`
  such that `a.equal_to(b)` for `Color` objects `a` and `b` returns `true` if and only if the RGB components are equal between `a` and `b`
- a member function
  `void invert();`
  to invert a color each RGB component `x` should be changed to `255-x`

Por exemplo:

Teste	Resultado

`Color c (1, 2, 3);`
`const Color& r = c;`
`cout << (int) r.red() << ' '`
`  << (int) r.green() << ' '`
`  << (int) r.blue() << ' '`
`  << boolalpha << r.equal_to(r) << '\n';`
	`1 2 3 true`

`cout << boolalpha`
`  << Color::WHITE.equal_to(Color::WHITE) << ' '`
`  << Color::BLACK.equal_to(Color::RED) << ' '`
`  << Color::BLUE.equal_to(Color::GREEN) << '\n';`
	`true false false`

`Color c(Color::WHITE);`
`cout << (int) c.red() << ' '`
`  << (int) c.green() << ' '`
`  << (int) c.blue() << ' '`
`  << boolalpha << c.equal_to(Color::WHITE) << ' '`
`  << c.equal_to(Color::BLACK) << '\n';`
	`255 255 255 true false`

`Color c(Color::WHITE);`
`c.invert();`
`cout << (int) c.red() << ' '`
`  << (int) c.green() << ' '`
`  << (int) c.blue() << ' '`
`  << boolalpha << c.equal_to(Color::WHITE) << ' '`
`  << c.equal_to(Color::BLACK) << '\n';`
	`0 0 0 false true`

Teste	Resultado

`Color c(255, 128, 12);`
`c.invert();`
`Color c2(c);`
`c2.invert();`
`cout << (int) c.red() << ' '`
`  << (int) c.green() << ' '`
`  << (int) c.blue() << ' '`
`  << boolalpha << c.equal_to({ 0, 127, 243 }) << ' '`
`  << (int) c2.red() << ' '`
`  << (int) c2.green() << ' '`
`  << (int) c2.blue() << ' '`
`  << boolalpha << c2.equal_to({ 255, 128, 12 }) << '\n';`
	`0 127 243 true 255 128 12 true`

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

Limpar resposta

```
// Submit your code using file attachments!
```

Color.cpp

Color.h

	Teste	Esperado	Recebido	

✔	`Color c (1, 2, 3);`
	`const Color& r = c;`
	`cout << (int) r.red() << ' '`
	`  << (int) r.green() << ' '`
	`  << (int) r.blue() << ' '`
	`  << boolalpha << r.equal_to(r) << '\n';`
		`1 2 3 true`	`1 2 3 true`	✔

✔	`cout << boolalpha`
	`  << Color::WHITE.equal_to(Color::WHITE) << ' '`
	`  << Color::BLACK.equal_to(Color::RED) << ' '`
	`  << Color::BLUE.equal_to(Color::GREEN) << '\n';`
		`true false false`	`true false false`	✔

✔	`Color c(Color::WHITE);`
	`cout << (int) c.red() << ' '`
	`  << (int) c.green() << ' '`
	`  << (int) c.blue() << ' '`
	`  << boolalpha << c.equal_to(Color::WHITE) << ' '`
	`  << c.equal_to(Color::BLACK) << '\n';`
		`255 255 255 true false`	`255 255 255 true false`	✔

✔	`Color c(Color::WHITE);`
	`c.invert();`
	`cout << (int) c.red() << ' '`
	`  << (int) c.green() << ' '`
	`  << (int) c.blue() << ' '`
	`  << boolalpha << c.equal_to(Color::WHITE) << ' '`
	`  << c.equal_to(Color::BLACK) << '\n';`
		`0 0 0 false true`	`0 0 0 false true`	✔

✔	`Color c(255, 128, 12);`
	`c.invert();`
	`Color c2(c);`
	`c2.invert();`
	`cout << (int) c.red() << ' '`
	`  << (int) c.green() << ' '`
	`  << (int) c.blue() << ' '`
	`  << boolalpha << c.equal_to({ 0, 127, 243 }) << ' '`
	`  << (int) c2.red() << ' '`
	`  << (int) c2.green() << ' '`
	`  << (int) c2.blue() << ' '`
	`  << boolalpha << c2.equal_to({ 255, 128, 12 }) << '\n';`
		`0 127 243 true 255 128 12 true`	`0 127 243 true 255 128 12 true`	✔

Passou em todos os testes! ✔

Solução do autor da pergunta (C):

```
/*
  // private tests (1000 points each)
```

Correta

Nota desta submissão: 20/20

Pergunta 2

Correta Pontuou 20 de 20

Consider the classes `Date` and `Person` whose declarations are given in `Date.h` and `Person.h`.

Write a C++ function that takes as input parameters a list of `Person` objects stored into a `vector` and a `Date` object, and shows on the screen the name and birthdate of all the persons that were born before the given date. The signature of the function is:

```
void born_before(const vector<Person>& persons, const Date& date)
```

Download the class declarations `Date.h` and `Person.h` and implement the code of the classes using the separate compilation principle, that is, with the definition of each class in a `.h` file and the implementation of its member functions in a corresponding `.cpp` file.

You must **submit four files**: `Date.h`, `Date.cpp`, `Person.h` and `Person.cpp`.

In your local workspace, to compile a program in a file containing the tests, named for instance `main.cpp`, you should use the supplied `Makefile` and execute the following command in the Terminal:

```
make PROG=main CPP_FILES="Date.cpp Person.cpp main.cpp" HEADERS="Date.h Person.h"
```

Por exemplo:

Teste	Resultado
`born_before({ {"Ana",{2000,4,5}}, {"Rui",{1999,5,11}}, {"Susana", {1999,5,13}}, {"Pedro",{2010,2,10}} }, {2000,1,1});`	`2000/1/1: Rui-1999/5/11 Susana-1999/5/13`
`born_before({ {"Rui",{2009,4,9}}, {"Susana",{1997,6,19}}, {"Pedro", {2018,3,10}} }, {2019,12,31});`	`2019/12/31: Rui-2009/4/9 Susana-1997/6/19 Pedro-2018/3/10`
`born_before({ {"Ana",{1999,5,12}}, {"Rui",{1960,3,21}}, {"Susana", {1999,7,25}}, {"Pedro",{1999,7,31}} }, {1970,1,1});`	`1970/1/1: Rui-1960/3/21`
`born_before({ {"Ana",{2001,7,15}}, {"Susana",{2019,8,12}}, {"Pedro", {2000,5,8}} }, {2001,1,1});`	`2001/1/1: Pedro-2000/5/8`
`born_before({ {"Pedro",{2000,11,7}} }, {2001,1,1});`	`2001/1/1: Pedro-2000/11/7`

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

Limpar resposta

```
#include <iostream>
#include <vector>
#include "Date.h"
#include "Person.h"

using namespace std;
void born_before(const vector<Person>& persons, const Date& date){
    date.show(); cout << ":";
    for(Person p: persons){
        if(p.get_birth_date().is_before(date)){
            cout << " "; cout << p.get_name() << "-"; p.get_birth_date().show();
        }
    }
    cout << endl;
}
```

Date.cpp

Date.h

Person.cpp

Person.h

	Teste	Esperado	Recebido	

	Teste	Esperado	Recebido	
✔	born_before({ {"Ana",{2000,4,5}}, {"Rui", {1999,5,11}}, {"Susana",{1999,5,13}}, {"Pedro", {2010,2,10}} }, {2000,1,1});	2000/1/1: Rui-1999/5/11 Susana-1999/5/13	2000/1/1: Rui-1999/5/11 Susana-1999/5/13	✔
✔	born_before({ {"Rui",{2009,4,9}}, {"Susana", {1997,6,19}}, {"Pedro",{2018,3,10}} }, {2019,12,31});	2019/12/31: Rui-2009/4/9 Susana-1997/6/19 Pedro-2018/3/10	2019/12/31: Rui-2009/4/9 Susana-1997/6/19 Pedro-2018/3/10	✔
✔	born_before({ {"Ana",{1999,5,12}}, {"Rui", {1960,3,21}}, {"Susana",{1999,7,25}}, {"Pedro", {1999,7,31}} }, {1970,1,1});	1970/1/1: Rui-1960/3/21	1970/1/1: Rui-1960/3/21	✔
✔	born_before({ {"Ana",{2001,7,15}}, {"Susana", {2019,8,12}}, {"Pedro",{2000,5,8}} }, {2001,1,1});	2001/1/1: Pedro-2000/5/8	2001/1/1: Pedro-2000/5/8	✔
✔	born_before({ {"Pedro",{2000,11,7}} }, {2001,1,1});	2001/1/1: Pedro-2000/11/7	2001/1/1: Pedro-2000/11/7	✔

Passou em todos os testes! ✔

Solução do autor da pergunta (C):

```
#include <iostream>
#include <vector>
#include "Date.h"
#include "Person.h"

using namespace std;

// persons born before date
void born_before(const vector<Person>& persons, const Date& date) {
  date.show(); cout << ": ";
  for (const auto &p : persons) {
    if (p.get_birth_date().is_before(date)) {
      p.show(); cout << ' ';
    }
  }
  cout << '\n';
}
```

Correta

Nota desta submissão: 20/20

Pergunta 3

Correta Pontuou 20 de 20

A polygon is made up of a set of vertices. Develop the following 2 classes to deal with polygons:

- `Point`: represents a point in 2D space with two integer attributes, `x` and `y`, which are the coordinates of the point.
- `Polygon`: represents the set of vertices of the polygon as a `vector` of objects of type `Point`.

You must decide which member functions of each class must be implemented and their signatures from the calls of the public tests. Other auxiliary functions may be necessary (for example, for calculating the distance between 2 points).

- For the users of class `Polygon`, the vertices are numbered starting with 1 (see the examples of usage of the member functions `get_vertex` and `add_vertex` in the tests).
- The functions `get_vertex` and `add_vertex` have as first parameter the number of the vertex to get from or to add to the polygon.
- For `get_vertex` this number must be in the range `[1 .. total_number_of_vertices]`.
- For `add_vertex` it must be in the range `[1 .. total_number_of_vertices+1]`; when a vertex is inserted in the middle of the set, the vertices that are after its position must be relocated to their new positions.
- The `perimeter` of a polygon is the sum of the length of its sides, which connect the points in order, with the last point connecting to the first.
- The `show` member function for `Point`'s must write the 2 coordinates between parenthesis and separated by a comma, for example: (3,1).
- The `show` member function for `Polygon`'s must write the coordinates of the vertices consecutively, between brackets, for example: {(1,1)(2,3)(0,1)}.
- The `show` member functions, both for `Point` and `Polygon`, must not write any end line character at the end of their output.
- Use the `const` qualifier in member functions and parameters whenever you find it adequate.
- The code must be written in separate files: Point.h and Polygon.h for the class declarations, and Point.cpp and Polygon.cpp, for the implementations.

In your local workspace, to compile a program in a file containing the tests, named for instance `main.cpp`, you should use the supplied `Makefile` and execute the following command in the Terminal:

```
make PROG=main CPP_FILES="Point.cpp Polygon.cpp main.cpp" HEADERS="Point.h Polygon.h"
```

Por exemplo:

Teste	Resultado
`Point p1, p2(0, 1);` `p1.show(); p2.show(); cout << '\n';`	`(0,0)(0,1)`

`Polygon poly1;`
`Point p1, p2(0, 1), p3(1, 0);`
`Polygon poly2(vector<Point>{ p1, p2, p3 });`
`poly1.show(); cout << " "; poly2.show(); cout << '\n';`
	`{} {(0,0)(0,1)(1,0)}`

`Point p1, p2(0, 1), p3(1, 0);`
`Polygon poly1(vector<Point>{ p1, p2, p3 });`
`cout << fixed << setprecision(3) << poly1.perimeter() << setprecision(0) << '\n';`
	`3.414`

`Point p1, p2(0, 1), p3(1, 0);`
`Polygon poly1(vector<Point>{ p1, p2, p3 });`
`Point p;`
`if (poly1.get_vertex(2, p)) { p.show(); cout << ' '; }`
`else cout << "vertex not found! ";`
`if (poly1.get_vertex(0, p)) { p.show(); cout << ' '; }`
`else cout << "vertex not found! ";`
`cout << '\n';`
	`(0,1) vertex not found!`

`Point p1, p2(0, 1), p3(1, 0), p4(1, 1);`
`Polygon poly1 = vector<Point>{ p1, p2, p3 };`
`poly1.add_vertex(3, p4);`
`poly1.show();`
`cout << ' ' << fixed << setprecision(3) << poly1.perimeter() << setprecision(0) << '\n';`
	`{(0,0)(0,1)(1,1)(1,0)}`
	`4.000`

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

Limpar resposta

```
// Submit your code using file attachments!
```

- Point.cpp
- Point.h
- Polygon.cpp
- Polygon.h

	Teste	Esperado	Recebido	
✔	`Point p1, p2(0, 1);` `p1.show(); p2.show(); cout << '\n';`	`(0,0)(0,1)`	`(0,0)(0,1)`	✔

✔	`Polygon poly1;`
	`Point p1, p2(0, 1), p3(1, 0);`
	`Polygon poly2(vector<Point>{ p1, p2, p3 });`
	`poly1.show(); cout << " "; poly2.show(); cout << '\n';`
		`{} {(0,0)(0,1)(1,0)}`	`{} {(0,0)(0,1)(1,0)}`	✔

✔	`Point p1, p2(0, 1), p3(1, 0);`
	`Polygon poly1(vector<Point>{ p1, p2, p3 });`
	`cout << fixed << setprecision(3) << poly1.perimeter() << setprecision(0) << '\n';`
		`3.414`	`3.414`	✔

✔	`Point p1, p2(0, 1), p3(1, 0);`
	`Polygon poly1(vector<Point>{ p1, p2, p3 });`
	`Point p;`
	`if (poly1.get_vertex(2, p)) { p.show(); cout << ' '; }`
	`else cout << "vertex not found! ";`
	`if (poly1.get_vertex(0, p)) { p.show(); cout << ' '; }`
	`else cout << "vertex not found! ";`
	`cout << '\n';`
		`(0,1) vertex not found!`	`(0,1) vertex not found!`	✔

✔	`Point p1, p2(0, 1), p3(1, 0), p4(1, 1);`
	`Polygon poly1 = vector<Point>{ p1, p2, p3 };`
	`poly1.add_vertex(3, p4);`
	`poly1.show();`
	`cout << ' ' << fixed << setprecision(3) << poly1.perimeter() << setprecision(0) << '\n';`
		`{(0,0)(0,1)(1,1)(1,0)} 4.000`	`{(0,0)(0,1)(1,1)(1,0)} 4.000`	✔

Passou em todos os testes! ✔

Solução do autor da pergunta (C):

```
/*
  // private tests (1000 points each)
```

Correta

Nota desta submissão: 20/20

Pergunta 4

Correta Pontuou 20 de 20

Write a C++ class template `Pair` that provides a way to store two heterogeneous objects as a single unit.

In the definition of the class template, include two data members, let us call them `first_` and `second_`, that can be of different types, and the following member functions:

- a constructor with parameters;
- `get_first()` and `get_second()` that return the `first_` and the `second_` data members, respectively;
- `show()` that shows the two elements of the pair, inside brackets and separated by a comma; for example: `{1,Porto}`, or `{A,65}`, or `{2000,366}`, depending on the type of elements of the pair; consider that the elements of the pair are either of a simple type (`int`, `double`, ...) or a C++-string

Consider now that you want to store a set of pairs into a `vector<Pair<string,int>>`, representing different type of data, for example the name and age of a set of persons, the name and grade of a set of students, or the name and population of a set of cities.

Write two functions, external to class `Pair`, `sort_by_first()` and `sort_by_second()`, that can sort the elements of a `vector<Pair<string,int>>` in non-descending order, taking into account the values of the `first_` or the `second_` attribute of the pairs, respectively.

- For example, if `v` is a `vector<Pair<string,int>>` that represents a set of names and ages of persons, the call `sort_by_second(v)` should order the elements of `v` by non-descending age.

Write also the function `show()`, external to class `Pair`, that shows on the screen the contents of a `vector<Pair<string,int>>`. See examples of the output of this function in the public tests.

Hint: the STL algorithm sort can be used to sort the elements of a vector `v` using the call `sort(v.begin(), v.end(), compare_func)` where `compare_func` is a function that, in this case, takes two arguments of type `Pair<string,int>` and returns `true` if the first argument is less than the second argument (i.e. it is ordered before).

Por exemplo:

Teste	Resultado

`vector<Pair<string, int>> persons = { {"Maria",17},{"Ana",21},{"Pedro",19} };`
`sort_by_first(persons);`
`show(persons); cout << '\n';`
	{{Ana,21}{Maria,17}{Pedro,19}}

`vector<Pair<string, int>> persons = { {"Ana",19},{"Rui",16} };`
`sort_by_second(persons);`
`show(persons); cout << '\n';`
	{{Rui,16}{Ana,19}}

`vector<Pair<string, int>> teams = { {"Porto",91},{"Benfica",74},`
`{"Sporting",85} };`
`sort_by_first(teams);`
`show(teams); cout << '\n';`
	{{Benfica,74}{Porto,91}
	{Sporting,85}}

`vector<Pair<string, int>> teams = { {"Porto",91},{"Benfica",74},`
`{"Sporting",85} };`
`sort_by_second(teams);`
`show(teams); cout << '\n';`
	{{Benfica,74}{Sporting,85}
	{Porto,91}}

`vector<Pair<string, int>> calories = { {"orange",37},{"egg",146},`
`{"apple",56},{"yogurt",51} };`
`sort_by_second(calories);`
`show(calories); cout << '\n';`
	{{orange,37}{yogurt,51}{apple,56}
	{egg,146}}

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

```
#include <iostream>
#include <string>
#include <algorithm>
#include <vector>

using namespace std;

template <typename T, typename V>
class Pair{
    public:
        Pair(T first, V second){
            first_ = first;
            second_ = second;
        }
        T get_first() const {return first_;}
        V get_second() const {return second_;}
        void show(){cout << '{' << first_ << ',' << second_ << '}';}
    private:
```

```
    private:
        T first_;
        V second_;
};

```

	Teste	Esperado	Recebido	
✔	`vector<Pair<string, int>> persons = { {"Maria",17}, {"Ana",21},{"Pedro",19} }; sort_by_first(persons); show(persons); cout << '\n';`	{{Ana,21}{Maria,17} {Pedro,19}}	{{Ana,21}{Maria,17} {Pedro,19}}	✔
✔	`vector<Pair<string, int>> persons = { {"Ana",19}, {"Rui",16} }; sort_by_second(persons); show(persons); cout << '\n';`	{{Rui,16}{Ana,19}}	{{Rui,16}{Ana,19}}	✔
✔	`vector<Pair<string, int>> teams = { {"Porto",91}, {"Benfica",74},{"Sporting",85} }; sort_by_first(teams); show(teams); cout << '\n';`	{{Benfica,74}{Porto,91} {Sporting,85}}	{{Benfica,74}{Porto,91} {Sporting,85}}	✔
✔	`vector<Pair<string, int>> teams = { {"Porto",91}, {"Benfica",74},{"Sporting",85} }; sort_by_second(teams); show(teams); cout << '\n';`	{{Benfica,74} {Sporting,85} {Porto,91}}	{{Benfica,74} {Sporting,85} {Porto,91}}	✔
✔	`vector<Pair<string, int>> calories = { {"orange",37},{"egg",146},{"apple",56},{"yogurt",51} }; sort_by_second(calories); show(calories); cout << '\n';`	{{orange,37}{yogurt,51} {apple,56}{egg,146}}	{{orange,37}{yogurt,51} {apple,56}{egg,146}}	✔

Passou em todos os testes! ✔

Solução do autor da pergunta (C):

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

using namespace std;

//! template class Pair
template <typename T1, typename T2>
class Pair {
public:
  Pair(const T1& first, const T2& second) : first_(first), second_(second) { }
  T1 get_first() const { return first_; }
  T2 get_second() const { return second_; }
  void show() const { std::cout << '{' << first_ << ',' << second_ << '}'; }
private:
  T1 first_;
  T2 second_;
};

//! Compare two pairs based on the value of first_
bool compare_first(const Pair<string, int>& p1, const Pair<string, int>& p2) {
```

Correta

Nota desta submissão: 20/20

Pergunta 5

Correta Pontuou 20 de 20

Consider the following template class `Stack<T>` defined in header `Stack.h`:

```
template <typename T>
struct node {
  T value;
  node<T>* next;
};

template <typename T>
class Stack {
public:
  Stack();
  ~Stack();
  size_t size() const;
  bool peek(T& elem) const;
  bool pop(T& elem);
  void push(const T& elem);
private:
  int size_;
  node<T>* top_;
};
```

The template is for a stack of elements stored using a singly-linked list of nodes using the `node` struct type. The stack should have the usual Last-In First-Out (LIFO) discipline: `push(v)` adds element `v` to the top of the stack, and `pop()` removes the element on top of the stack, i.e., the one that has been added to the stack most recently through `push()`.

The class should work as follows:

- The `top_` member field should point to the top of the stack, and the `size_` member field should indicate the total number of elements in the stack;
- `Stack()` builds an initially empty stack;
- `~Stack()` releases the associated memory to the stack elements;
- `size()` returns the number of elements stored in the stack;
- `push(v)` adds an element to the (top of the) stack;
- `pop(v)`:
  - if the stack is not empty, removes the element on top of the stack, assigns that element to `v`, and returns `true`; or
  - simply returns `false` if the stack is empty, leaving `v` unchanged.
- `peek(v)`:
  - if the stack is not empty, assigns `v` with the element on top of the stack *without removing the element* and returns `true`; or
  - simply returns `false` if the stack is empty, leaving `v` unchanged.

You should properly use `new` and `delete` for node allocation and release in order to avoid memory errors such as leaks, dangling references, etc.

Por exemplo:

Teste	Resultado

`Stack<int> s;`
`const Stack<int>& r = s;`
`int v = -1;`
`cout << r.size()  << ' ' << boolalpha`
`  << r.peek(v) << ' ' << v << ' '`
`  << s.pop(v)  << ' ' << v << ' '`
`  << s.size() << '\n';`
	`0 false -1 false -1 0`

`Stack<int> s;`
`int v = -1;`
`s.push(123);`
`cout << s.size()  << ' ' << boolalpha`
`  << s.peek(v) << ' ' << v << ' '`
`  << s.pop(v)  << ' ' << v << ' '`
`  << s.size() << '\n';`
	`1 true 123 true 123 0`

`Stack<string> s; string v;`
`s.push("a"); s.push("b"); s.push("c");`
`cout << s.size();`
`while(s.pop(v)) cout << ' ' << v;`
`cout << ' ' << s.size() << '\n';`
	`3 c b a 0`

Teste	Resultado

`Stack<int> s; int v = -1;`
`s.push(111); s.push(222); s.push(333);`
`cout << s.size();`
`while(s.peek(v)) {`
`  cout << ' ' << v;`
`  s.pop(v);`
`  cout << ' ' << v;`
`  if (v % 2 != 0) s.push(v + 1);`
`  cout << ' ' << s.size(); }`
`cout << '\n';`
	`3 333 333 3 334 334 2 222 222 1 111 111 1 112 112 0`

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

Limpar resposta

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

#include "Stack.h"

template <typename T>
Stack<T>::Stack(){
    size_ = 0;
    top_ = nullptr;
}

template <typename T>
Stack<T>::~Stack(){
    while(top_ != nullptr){
        size_--;
        node<T>* aux = top_->next;
        delete top_;
        top_ = aux;
    }
}

```

	Teste	Esperado	Recebido	

✔	`Stack<int> s;`
	`const Stack<int>& r = s;`
	`int v = -1;`
	`cout << r.size()  << ' ' <<`
	`boolalpha`
	`  << r.peek(v) << ' ' <<`
	`v << ' '`
	`  << s.pop(v)  << ' ' <<`
	`v << ' '`
	`  << s.size() << '\n';`
		`0 false -1 false -1 0`	`0 false -1 false -1 0`	✔

✔	`Stack<int> s;`
	`int v = -1;`
	`s.push(123);`
	`cout << s.size()  << ' ' <<`
	`boolalpha`
	`  << s.peek(v) << ' ' <<`
	`v << ' '`
	`  << s.pop(v)  << ' ' <<`
	`v << ' '`
	`  << s.size() << '\n';`
		`1 true 123 true 123 0`	`1 true 123 true 123 0`	✔

✔	`Stack<string> s; string v;`
	`s.push("a"); s.push("b");`
	`s.push("c");`
	`cout << s.size();`
	`while(s.pop(v)) cout << ' '`
	`<< v;`
	`cout << ' ' << s.size() <<`
	`'\n';`
		`3 c b a 0`	`3 c b a 0`	✔

	Teste	Esperado	Recebido	

✔	`Stack<int> s; int v = -1;`
	`s.push(111); s.push(222);`
	`s.push(333);`
	`cout << s.size();`
	`while(s.peek(v)) {`
	`  cout << ' ' << v;`
	`  s.pop(v);`
	`  cout << ' ' << v;`
	`  if (v % 2 != 0) s.push(v + 1);`
	`  cout << ' ' << s.size();`
	`}`
	`cout << '\n';`
		3 333 333 3 334 334 2 222 222 1 111 111 1 112 112 0	3 333 333 3 334 334 2 222 222 1 111 111 1 112 112 0	✔

Passou em todos os testes! ✔

Solução do autor da pergunta (C):

```
#include <iostream>
#include <iomanip>
using namespace std;
#include "Stack.h"

template <typename T>
Stack<T>::Stack() : size_(0), top_(nullptr) { }

template <typename T>
Stack<T>::~Stack() {
  node<T>* n = top_;
  while (n != nullptr) {
    node<T>* aux = n->next;
    delete n;
    n = aux;
  }
}

template <typename T>
size_t Stack<T>::size() const {
  return size_;
}
```

Correta

Nota desta submissão: 20/20

◄ T08 10/05

Ir para...

T09 17/05 ►