Câu hỏi 1

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Cho một **struct** mô tả một số phức

```
struct Complex { double a, b; };
```

Trong đó a là phần thực và b là phần ảo của số phức. Viết các hàm sau:

- `void print(const Complex& c);` nhận đầu vào là một số phức, in ra số phức dưới dạng $a + bi$ nếu $b > 0$ hoặc $a - bi$ nếu $b < 0$. Nếu $b = 1$ hoặc $b = -1$ thì không in ra b. Nếu a hoặc b bằng 0 thì lược bỏ phần có chứa nó. Nếu cả a và b bằng 0 thì in ra 0. In thêm dấu xuống dòng ở cuối.
- `Complex add(const Complex& c1, const Complex &c2);` nhận từ đầu vào 2 số phức và trả về một số phức là tổng của 2 số đó.
- `int compare(const Complex& c1, const Complex &c2);` nhận từ đầu vào 2 số phức và trả về 0 nếu $|c1| == |c2|$, trả về 1 nếu $|c1| > |c2|$ và -1 nếu $|c1| < |c2|$.

For example:

Test	Result

`Complex c1 = {1, 3};`
`Complex c2 = {2, 1};`
`print(add(c1, c2));`
`cout << compare(c1, c2);`
	`3+4i`
	`1`

Answer: (penalty regime: 0 %)

```
void print(const Complex &c) {
  if (c.a == 0 && c.b == 1) {
    cout << "i";
  } else if (c.a == 0 && c.b == 0) {
    cout << 0;
  } else if (c.b == 0) {
    cout << c.a;
  } else {
    if (c.a) {
      cout << c.a;
    }
    cout << (c.b < 0? '-' : '+');
    if (abs(c.b) != 1) {
      cout << abs(c.b);
    }
    cout << 'i';
  }
  cout << '\n';
}

Complex add(const Complex &c1, const Complex &c2) {
  Complex c;
  c.a = c1.a + c2.a;
  c.b = c1.b + c2.b;
  return c;
}

int compare(const Complex &c1, const Complex &c2) {
  double m1 = c1.a * c1.a + c1.b * c1.b;
  double m2 = c2.a * c2.a + c2.b * c2.b;
  if (m1 == m2) return 0;
  if (m1 > m2) return 1;
  return -1;
}
```

	Test	Expected	Got	

✓	Complex c1 = {1, 3};
	Complex c2 = {2, 1};
	print(add(c1, c2));
	cout << compare(c1, c2);
		3+4i
		1
			3+4i
			1
				✓

✓	Complex c1 = {-1, 0};
	Complex c2 = {1, -2};
	Complex c3 = {0, 1};
	Complex c4 = {0, 0};
	Complex c5 = {2, 1};
	print(c1);
	print(c2);
	print(c3);
	print(c4);
	print(c5);
	cout << compare(c1, c2) << endl;
	cout << compare(c3, c4) << endl;
	cout << compare(c5, add(c5, c4)) << endl;
	print(add(c1, add(c2, c3)));
		-1
		1-2i
		i
		0
		2+i
		-1
		1
		0
		-i
			-1
			1-2i
			i
			0
			2+i
			-1
			1
			0
			-i
				✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 2

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Cho một **struct** mô tả một bức ảnh như sau:

`struct Image { int *pixels; int width; int height; }`

Trong đó $width$ và $height$ lần lượt là chiều rộng và chiều cao bức ảnh, $pixels$ là mảng chứa $width * height$ giá trị, $pixels[i * width + j]$ thể hiện giá trị điểm ảnh hàng i cột j. Viết các hàm sau:

- `void printImage(const Image& img);` nhận đầu vào là một ảnh và in ra màn hình giá trị các điểm ảnh của bức ảnh img thành một bảng $n \times m$ với n và m là chiều cao và chiều rộng của ảnh. Các giá trị cách nhau một dấu cách. Hàm không làm thay đổi ảnh đầu vào.
- `Image halve(const Image& img);` nhận đầu vào là một ảnh và trả về một ảnh có kích thước mỗi chiều bằng một nửa ảnh đầu vào. Ảnh trả về tạo ra bằng cách loại bỏ các hàng và cột có chỉ số lẻ trong ảnh đầu vào. Hàm không làm thay đổi ảnh đầu vào.

For example:

Test	Result

`int pixels[] = {`
`  1, 3, 0, 5, 2,`
`  4, 1, 8, 3, 1,`
`  3, 3, 1, 3, 2`
`};`
`Image img = {pixels, 5, 3};`
`printImage(img);`
`printImage(halve(img));`
	`1 3 0 5 2`
	`4 1 8 3 1`
	`3 3 1 3 2`
	`1 0 2`
	`3 1 2`

Answer: (penalty regime: 0 %)

```
void printImage(const Image& img) {
  for (int i = 0; i < img.height; ++i) {
    for (int j = 0; j < img.width; ++j) {
      cout << img.pixels[i * img.width + j] << ' ';
    }
    cout << '\n';
  }
}

Image halve(const Image& img) {
  Image res;
  res.width = (img.width + 1) / 2;
  res.height = (img.height + 1) / 2;
  res.pixels = new int[res.width * res.height];
  int k = 0;
  for (int i = 0; i < img.height; i += 2) {
    for (int j = 0; j < img.width; j += 2) {
      res.pixels[k++] = img.pixels[i * img.width + j];
    }
  }
  return res;
}

```

	Test	Expected	Got	

✓	int pixels[] = {
	    1, 3, 0, 5, 2,
	    4, 1, 8, 3, 1,
	    3, 3, 1, 3, 2
	};
	Image img = {pixels, 5, 3};
	printImage(img);
	printImage(halve(img));
		1 3 0 5 2
		4 1 8 3 1
		3 3 1 3 2
		
		1 0 2
		3 1 2
			1 3 0 5 2
			4 1 8 3 1
			3 3 1 3 2
			
			1 0 2
			3 1 2
				✓

✓	int pixels[] = {
	    1, 2, 3, 4, 5, 6, 7, 8, 9,
	    5, 4, 1, 2, 5, 6, 8, 2, 1,
	    8, 6, 2, 2, 1, 6, 8, 9, 0,
	    1, 0, 9, 9, 9, 9, 9, 9, 0,
	    6, 7, 8, 5, 6, 7, 8, 9, 2
	};
	Image img = {pixels, 9, 5};
	printImage(img);
	printImage(halve(img));
		1 2 3 4 5 6 7 8 9
		5 4 1 2 5 6 8 2 1
		8 6 2 2 1 6 8 9 0
		1 0 9 9 9 9 9 9 0
		6 7 8 5 6 7 8 9 2
		
		1 3 5 7 9
		8 2 1 8 0
		6 8 6 8 2
			1 2 3 4 5 6 7 8 9
			5 4 1 2 5 6 8 2 1
			8 6 2 2 1 6 8 9 0
			1 0 9 9 9 9 9 9 0
			6 7 8 5 6 7 8 9 2
			
			1 3 5 7 9
			8 2 1 8 0
			6 8 6 8 2
				✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 3

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Cho một **struct** mô tả một sinh viên như sau:

`struct Student { string id; string name; double gpa; }`

Trong đó id là mã số sinh viên, $name$ là tên sinh viên, gpa là điểm trung bình của sinh viên. Viết các hàm sau:

- `void print(const vector<Student>& student_list);` nhận đầu vào là danh sách các sinh viên, in ra màn hình thông tin các sinh viên trong danh sách, mỗi sinh viên một dòng. Thông tin in ra là các giá trị id, $name$ và gpa cách nhau bởi 1 dấu cách. Hàm không làm thay đổi danh sách ban đầu.
- `int find(const vector<Student>& student_list, string id);` nhận đầu vào là một danh sách các sinh viên và mã số của sinh viên cần tìm. Hàm trả về chỉ số của sinh viên có mã số là id trong danh sách sinh viên $student\_list$, trả về -1 nếu không nằm trong danh sách. Hàm không làm thay đổi danh sách ban đầu.
- `vector<Student> top3(const vector<Student>& student_list);` nhận đầu vào là một danh sách các sinh viên và trả về danh sách gồm tối đa 3 sinh viên có gpa cao nhất, sắp xếp giảm dần theo gpa. Hàm không làm thay đổi danh sách ban đầu.

For example:

Test	Result

`vector<Student> students = {`
`{"1", "Le Quang Duy", 5.5},`
`{"2", "Nguyen Tan Dat", 6},`
`{"10", "Cao Duy Manh", 3},`
`{"4", "Nguyen Van Ngoc", 4.5},`
`{"3", "Trieu Dinh Nguyen", 4}`
`};`
`vector<Student> top_students = top3(students);`
`print(top_students);`
`cout << find(students, "10") << endl;`
`cout << find(students, "11") << endl;`
	`2 Nguyen Tan Dat 6`
	`1 Le Quang Duy 5.5`
	`4 Nguyen Van Ngoc 4.5`
	`2`
	`-1`

Answer: (penalty regime: 0 %)

```
void print(const vector<Student>& student_list) {
  for (const Student &s : student_list) {
    cout << s.id << ' ' << s.name << ' ' << s.gpa << '\n';
  }
}

int find(const vector<Student>& student_list, string id) {
  for (size_t i = 0; i < student_list.size(); ++i) {
    if (student_list[i].id == id) {
      return i;
    }
  }
  return -1;
}

vector<Student> top3(const vector<Student>& student_list) {
  size_t sz = min(student_list.size(), size_t(3));
  vector<Student> cpy = student_list;
  partial_sort(cpy.begin(), cpy.end(), cpy.begin() + sz, [&](const Student &a, const Student &b) {
    return a.gpa > b.gpa;
  });
  return vector<Student>(cpy.begin(), cpy.begin() + sz);
}
```

	Test	Expected	Got	

✓	vector<Student> students = {
	{"1", "Le Quang Duy", 5.5},
	{"2", "Nguyen Tan Dat", 6},
	{"10", "Cao Duy Manh", 3},
	{"4", "Nguyen Van Ngoc", 4.5},
	{"3", "Trieu Dinh Nguyen", 4}
	};
	vector<Student> top_students = top3(students);
	print(top_students);
	cout << find(students, "10") << endl;
	cout << find(students, "11") << endl;
		2 Nguyen Tan Dat 6
		1 Le Quang Duy 5.5
		4 Nguyen Van Ngoc 4.5
		2
		-1
			2 Nguyen Tan Dat 6
			1 Le Quang Duy 5.5
			4 Nguyen Van Ngoc 4.5
			2
			-1
				✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi **4**

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Dựng lớp $MyFraction$ biểu diễn phân số với các hàm tính toán sau:

```
class MyFraction {<br>private:<br>    int a, b;<br>public:<br>    MyFraction(int _a = 1, int _b = 1);<br>    void print() const;<br>    MyFraction add(const MyFraction& f) const;<br>    MyFraction substract(const MyFraction& f) const;<br>MyFraction multiply(const MyFraction& f) const;<br>    MyFraction divide(const MyFraction& f) const;<br>    void simplify();<br>    int compare(const MyFraction& f) const;<br>};
```

Trong đó:

- Hàm `MyFraction add(const MyFraction& f) const` trả về kết quả là tổng của phân số hiện tại với phân số **f**. Hàm này không thay đổi giá trị của phân số hiện tại.
- Hàm `MyFraction substract(const MyFraction& f) const` trả về kết quả là hiệu của phân số hiện tại với phân số **f**. Hàm này không thay đổi giá trị của phân số hiện tại.
- Hàm `MyFraction multiply(const MyFraction& f) const` trả về kết quả là tích của phân số hiện tại với phân số **f**. Hàm này không thay đổi giá trị của phân số hiện tại.
- Hàm `MyFraction divide(const MyFraction& f) const` trả về kết quả là thương của phân số hiện tại với phân số **f**. Hàm này không thay đổi giá trị của phân số hiện tại.
- Hàm `void simplify()` tối giản phân số hiện tại.
- Hàm `int compare(const MyFraction& f) const` trả về 1 nếu phân số hiện tại lớn hơn phân số **f**; trả về -1 nếu phân số hiện tại nhỏ hơn phân số **f**; trả về 0 nếu bằng nhau.
- Hàm `void print() const` in ra phân số dưới dạng a/b nếu $b \neq 0$. Ngược lại in ra $invalid$. Lưu ý: cần tối giản phân số trước khi in ra kết quả.

For example:

Test	Result

`MyFraction x(1, 3), y(1, 3);`
`x = x.add(y);`
`x.print();`
	2/3

Answer: (penalty regime: 0 %)

Reset answer

```
class MyFraction {
private:
  int a, b;
public:
  MyFraction(int _a = 1, int _b = 1);
  void print() const;
  MyFraction add(const MyFraction &f) const;
  MyFraction substract(const MyFraction &f) const;
  MyFraction multiply(const MyFraction &f) const;
  MyFraction divide(const MyFraction &f) const;
  void simplify();
  int compare(const MyFraction &f) const;
};

MyFraction::MyFraction(int _a, int _b) {
  a = _a;
  b = _b;
}

MyFraction MyFraction::add(const MyFraction &f) const {
  MyFraction c(a * f.b + b * f.a, b * f.b);
  c.simplify();
  return c;
}

```

```cpp
MyFraction MyFraction::substract(const MyFraction &f) const {
  MyFraction c(a * f.b - b * f.a, b * f.b);
  c.simplify();
  return c;
}

MyFraction MyFraction::multiply(const MyFraction &f) const {
  MyFraction c(a * f.a, b * f.b);
  c.simplify();
  return c;
}

MyFraction MyFraction::divide(const MyFraction &f) const {
  MyFraction c(a * f.b, b * f.a);
  c.simplify();
  return c;
}

void MyFraction::simplify() {
  bool neg = (a < 0) ^ (b < 0);
  a = abs(a);
  b = abs(b);
  int g = __gcd(a, b);
  a /= g;
  b /= g;
  if (neg) {
    a = -a;
```

	Test	Expected	Got	

✓	`MyFraction x(1, 3), y(1, 3);`
	`x = x.add(y);`
	`x.print();`
		2/3	2/3	✓

✓	`MyFraction x, y(-1, 2);`
	`x = x.substract(y);`
	`x.print();`
		3/2	3/2	✓

✓	`MyFraction x(4, 5), y(5, 8);`
	`x = x.multiply(y);`
	`x.print();`
		1/2	1/2	✓

✓	`MyFraction x(3, 4), y(4, 3), z(2, 3);`
	`x = x.multiply(y.substract(z));`
	`x.print();`
		1/2	1/2	✓

✓	`MyFraction x(4, 5), y(5, 20), z;`
	`MyFraction result = (x.multiply(y)).divide(z);`
	`result.print();`
		1/5	1/5	✓

✓	`MyFraction x(1, 0);` `x.print();`	invalid	invalid	✓

✓	`MyFraction x(4, 5), y(2, 5), z(-3, 5);`
	`MyFraction result1 = x.add(y);`
	`MyFraction result2 = y.substract(z);`
	`if (result1.compare(result2) == 1) {`
	`  cout << "result1 > result2";`
	`} else {`
	`  cout << "result1 <= result2";`
	`}`
		result1 > result2	result1 > result2	✓

	Test	Expected	Got	

✓	MyFraction x(4, 5), y(2, 5), z(-8, 5);
	MyFraction result1 = x.add(y);
	MyFraction result2 = y.substract(z);
	if (result1.compare(result2) == -1) {
	    cout << "result1 < result2";
	} else {
	    cout << "result1 >= result2";
	}
		result1 < result2	result1 < result2	✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi **5**

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Dựng lớp **Vector** thể hiện **vector 2 chiều** với các hàm tính toán trên vector như sau

```
class Vector {
    double x_, y_;
public:
    Vector(double x = 0, double y = 0);
    void print(int precision = 2, bool newLine = true);
    void truncate(double length);
    bool isOrtho(const Vector& v) const;
    static float dot(const Vector& v1, const Vector& v2);
    static float cross(const Vector& v1, const Vector& v2);
    friend ostream& operator<<(ostream& os, const Vector& v);
};
```

Trong đó:

- Hàm `void truncate(double length)` giảm độ dài các tọa độ của vecto **v** đi một khoảng **length**. Hàm này làm thay đổi tọa độ của vecto hiện tại.
- Hàm `bool isOrtho(const Vector& v) const` kiểm tra xem vecto hiện tại và vecto **v** có trực giao hay không. Nếu có, hàm trả về giá trị $true$, ngược lại, trả về giá trị $false$. Hàm không làm thay đổi giá trị của vector hiện tại.
- Hàm `static float dot(const Vector& v1, const Vector& v2)` thực hiện phép nhân vô hướng hai vector, hàm trả về giá trị là một số thực. Hàm không làm thay đổi giá trị của hai vector đầu vào.
- Hàm `static float cross(const Vector& v1, const Vector& v2)` tính tích chéo của hai vector **v1** và **v2.** Hàm trả về kết quả tích chéo là một số thực. Hàm không làm thay đổi giá trị của hai vector đầu vào.
- Hàm `friend ostream& operator<<(ostream& os, const Vector& v)` thực hiện việc định nghĩa lại toán tử $<<$ thực hiện việc in kết quả của một vector ra màn hình, kết quả tương tự như khi thực hiện hàm `Vector::print()`. Hàm trả về luồng đầu ra được định nghĩa tương ứng và không làm thay đổi giá trị của vector truyền vào.

For example:

Test	Result

`Vector y(2, 4);`
`y.truncate(2);`
`y.print();`
	`(0.00, 2.00)`

Answer: (penalty regime: 0 %)

```
class Vector {
  double x_, y_;
public:
  Vector(double x = 0, double y = 0);
  void print(int precision = 2, bool newLine = true);
  void truncate(double length);
  bool isOrtho(const Vector &v) const;
  static float dot(const Vector &v1, const Vector &v2);
  static float cross(const Vector &v1, const Vector &v2);
  friend ostream &operator<<(ostream &os, const Vector &v);
};

Vector::Vector(double x, double y) {
  x_ = x;
  y_ = y;
}

void Vector::print(int precision, bool newLine) {
  cout << fixed << setprecision(precision);
```

```
  if (newLine) {
    cout << '\n';
  }
}

void Vector::truncate(double length) {
  x_ -= length;
  y_ -= length;
}

bool Vector::isOrtho(const Vector &v) const {
  return dot(*this, v) == 0;
}

float Vector::dot(const Vector &v1, const Vector &v2) {
  return v1.x_ * v2.x_ + v1.y_ * v2.y_;
}

float Vector::cross(const Vector &v1, const Vector &v2) {
  return v1.x_ * v2.y_ - v1.y_ * v2.x_;
}

ostream& operator<<(ostream &os, const Vector &v) {
  os << '(' << v.x_ << ", " << v.y_ << ')';
  return os;
}
```

	Test	Expected	Got	

✓	Vector y(2, 4);
	y.truncate(2);
	y.print();
		(0.00, 2.00)	(0.00, 2.00)	✓

✓	Vector x(-2,1), y(2, 4); cout << boolalpha << x.isOrtho(y);	true	true	✓
✓	Vector x(0,0), y(2, 4); cout << boolalpha << x.isOrtho(y);	true	true	✓
✓	Vector x(-2,12), y(2, 4); cout << fixed << setprecision(2) << Vector::dot(x, y);	44.00	44.00	✓
✓	Vector x(2,12), y(2, 4); cout << fixed << setprecision(2) << Vector::cross(x, y);	-16.00	-16.00	✓
✓	Vector x(-2,12); cout << x;	(-2.00, 12.00)	(-2.00, 12.00)	✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 6

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Dựng lớp **Vector** thể hiện **vector 2 chiều** với các hàm tính toán trên vector như sau

```
class Vector {
    double x_, y_;
public:
    Vector(double x = 0, double y = 0);
    void print(int precision = 2, bool newLine = true);
    Vector& operator=(const Vector& v);
    Vector operator+(const Vector& v) const;
    Vector operator-(const Vector& v) const;
    Vector& operator+=(const Vector& v);
    Vector& operator-=(const Vector& v);
};
```

Trong đó:

- Hàm `Vector& operator=(const Vector& v)` thay đổi giá trị vector hiện tại cho giống với vector **v** và trả lại tham chiếu đến vector hiện tại.
- Hàm `Vector operator+(const Vector& v) const` trả lại kết quả là vector tổng của vector hiện tại với vector **v**. Hàm này không thay đổi giá trị vector hiện tại.
- Hàm `Vector operator-(const Vector& v) const` trả lại kết quả là vector hiệu của vector hiện tại với vector **v**. Hàm này không thay đổi giá trị vector hiện tại.
- Hàm `Vector& operator+=(const Vector& v)` cộng thêm vector **v** vào vector hiện tại (thay đổi nó) và trả lại tham chiếu đến vector hiện tại.
- Hàm `Vector& operator-=(const Vector& v)` trừ vector hiện tại (thay đổi nó) đi một lượng là vector **v** và trả lại tham chiếu đến vector hiện tại.

Các phần đã làm ở bài trước có thể sao chép sang bài này.

For example:

Test	Result

`Vector x(1,2), y(3, 3), z;`
`z = x + y;`
`z.print()`
	`(4.00, 5.00)`

Answer: (penalty regime: 0 %)

```
class Vector {
  double x_, y_;
public:
  Vector(double x = 0, double y = 0);
  void print(int precision = 2, bool newLine = true);
  Vector &operator=(const Vector &v);
  Vector operator+(const Vector &v) const;
  Vector operator-(const Vector &v) const;
  Vector &operator+=(const Vector &v);
  Vector &operator-=(const Vector &v);
};

Vector::Vector(double x, double y) {
  x_ = x;
  y_ = y;
}

void Vector::print(int precision, bool newLine) {
  cout << fixed << setprecision(precision);
  cout << '(' << x_ << ", " << y_ << ')';
```

```
  if (newLine) {
    cout << '\n';
  }
}

Vector &Vector::operator=(const Vector &v) {
  x_ = v.x_;
  y_ = v.y_;
  return *this;
}

Vector Vector::operator+(const Vector &v) const {
  return Vector(x_ + v.x_, y_ + v.y_);
}

Vector Vector::operator-(const Vector &v) const {
  return Vector(x_ - v.x_, y_ - v.y_);
}

Vector &Vector::operator+=(const Vector &v) {
  x_ += v.x_;
  y_ += v.y_;
  return *this;
}

Vector &Vector::operator-=(const Vector &v) {
  x_ -= v.x_;
  y_ -= v.y_;
  return *this;
}
```

	Test	Expected	Got	

✓	Vector x, y(2, 3);
	x = y;
	x.print();
		(2.00, 3.00)	(2.00, 3.00)	✓

✓	Vector x(1,2), y(3, 3), z; z = x + y; z.print()	(4.00, 5.00)	(4.00, 5.00)	✓
✓	Vector x(3, 2), y(1, 1), z; z = x - y; z.print()	(2.00, 1.00)	(2.00, 1.00)	✓
✓	Vector x(1,2), y(3, 3); x += y; x.print()	(4.00, 5.00)	(4.00, 5.00)	✓
✓	Vector x(3, 2), y(1, 1); x -= y; x.print()	(2.00, 1.00)	(2.00, 1.00)	✓

✓	Vector x, y(1, 2), z(4, 2), w;
	x = z;
	x += y;
	w = x + z;
	w.print();
		(9.00, 6.00)	(9.00, 6.00)	✓

✓	Vector x(1, 2), y(3, 1), z(2, 2), w; w = x + y + z; w.print()	(6.00, 5.00)	(6.00, 5.00)	✓
✓	Vector x(6, 6), y(1, 1), z(1, 2), w; w = x - y - z; w.print()	(4.00, 3.00)	(4.00, 3.00)	✓

	Test	Expected	Got	

✓	`Vector x(3, 2), y(1, 1), z(2, 2);`
	`z += x - y;`
	`z.print()`
		(4.00, 3.00)	(4.00, 3.00)	✓

✓	`Vector x(3, 2), y(1, 1), z(6, 6);`
	`z -= x + y;`
	`z.print()`
		(2.00, 3.00)	(2.00, 3.00)	✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 7

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Dựng lớp **Vector** thể hiện **vector 2 chiều** với các hàm tính toán trên vector như sau

```
class Vector {
    double x_, y_;
public:
    Vector(double x = 0, double y = 0);
    void print(int precision = 2, bool newLine = true);
    Vector operator+(double s) const;
    Vector operator-(double s) const;
    Vector operator*(double s) const;
    Vector operator/(double s) const;
    Vector& operator+=(double s);
    Vector& operator-=(double s);
    Vector& operator*=(double s);
    Vector& operator/=(double s);
};
```

Trong đó:

- Hàm `Vector operator+(double s) const` trả lại kết quả là vector tổng của vector hiện tại với 1 số **s** (cộng cả x_ và y_ với s). Hàm này không thay đổi giá trị vector hiện tại.
- Hàm `Vector operator-(double s) const` trả lại kết quả là vector hiệu của vector hiện tại với 1 số **s** (trừ cả x_ và y_ với s). Hàm này không thay đổi giá trị vector hiện tại.
- Hàm `Vector operator*(double s) const` trả lại kết quả là vector tích của vector hiện tại với 1 số **s** (nhân cả x_ và y_ với s). Hàm này không thay đổi giá trị vector hiện tại.
- Hàm `Vector operator/(double s) const` trả lại kết quả là vector thương của vector hiện tại với 1 số **s** (chia cả x_ và y_ với s). Hàm này không thay đổi giá trị vector hiện tại.
- Hàm `Vector& operator+=(double s)` cộng thêm số **s** vào vector hiện tại (thay đổi nó) và trả lại tham chiếu đến vector hiện tại.
- Hàm `Vector& operator-=(double s)` trừ đi số **s** vào vector hiện tại (thay đổi nó) và trả lại tham chiếu đến vector hiện tại.
- Hàm `Vector& operator*=(double s)` nhân thêm số **s** vào vector hiện tại (thay đổi nó) và trả lại tham chiếu đến vector hiện tại.
- Hàm `Vector& operator/=(double s)` chia cho số **s** vào vector hiện tại (thay đổi nó) và trả lại tham chiếu đến vector hiện tại.

Các phần đã làm ở bài trước có thể sao chép sang bài này.

For example:

Test	Result

`Vector x(2, 3);`
`x += 2;`
`x.print();`
	`(4.00, 5.00)`

Answer: (penalty regime: 0 %)

```
1  class Vector {
2    double x_, y_;
3  public:
4    Vector(double x = 0, double y = 0);
5    void print(int precision = 2, bool newLine = true);
6    Vector operator+(double s) const;
```

```
  Vector operator-(double s) const;
  Vector operator*(double s) const;
  Vector operator/(double s) const;
  Vector &operator+=(double s);
  Vector &operator-=(double s);
  Vector &operator*=(double s);
  Vector &operator/=(double s);
};

Vector::Vector(double x, double y) {
  x_ = x;
  y_ = y;
}

void Vector::print(int precision, bool newLine) {
  cout << fixed << setprecision(precision);
  cout << '(' << x_ << ", " << y_ << ')';
  if (newLine) {
    cout << '\n';
  }
}

Vector Vector::operator+(double s) const {
  return Vector(x_ + s, y_ + s);
}

Vector Vector::operator-(double s) const {
  return Vector(x_ - s, y_ - s);
}

Vector Vector::operator*(double s) const {
  return Vector(x_ * s, y_ * s);
}

Vector Vector::operator/(double s) const {
  return Vector(x_ / s, y_ / s);
}

Vector &Vector::operator+=(double s) {
  x_ += s;
  y_ += s;
  return *this;
}

Vector &Vector::operator-=(double s) {
  x_ -= s;
```

	Test	Expected	Got	

✓	`Vector x, y(2, 3);`
	`x = y + 2;`
	`x.print();`
		`(4.00, 5.00)`	`(4.00, 5.00)`	✓

✓	`Vector x, y(2, 3);`
	`x = y - 2;`
	`x.print();`
		`(0.00, 1.00)`	`(0.00, 1.00)`	✓

✓	`Vector x, y(2, 3);`
	`x = y * 2;`
	`x.print();`
		`(4.00, 6.00)`	`(4.00, 6.00)`	✓

✓	`Vector x, y(2, 3);`
	`x = y / 2;`
	`x.print();`
		`(1.00, 1.50)`	`(1.00, 1.50)`	✓

✓	`Vector x(2, 3);`
	`x += 2;`
	`x.print();`
		`(4.00, 5.00)`	`(4.00, 5.00)`	✓

	Test	Expected	Got	

✓	`Vector x(2, 3);`
	`x -= 2;`
	`x.print();`
		`(0.00, 1.00)`	`(0.00, 1.00)`	✓

✓	`Vector x(2, 3);`
	`x *= 2;`
	`x.print();`
		`(4.00, 6.00)`	`(4.00, 6.00)`	✓

✓	`Vector x(2, 3);`
	`x /= 2;`
	`x.print();`
		`(1.00, 1.50)`	`(1.00, 1.50)`	✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 8

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Dựng lớp Fraction biểu diễn phân số với các hàm tính toán sau:

```
class Fraction {
    int a, b;
public:
    Fraction(int a = 1, int b = 1);
    friend ostream& operator<<(ostream& os, const Fraction& f);
    Fraction operator+(const Fraction& f) const;
    Fraction operator-(const Fraction& f) const;
    Fraction operator*(const Fraction& f) const;
    Fraction operator/(const Fraction& f) const;
    <span data-sheets-value="{"1":2,"2":"void simplify()"}" data-sheets-userformat="{"2":513,"3":[null,0],"12":0}">void
simplify();
    bool operator>(const Fraction& f) const;
    bool operator<(const Fraction& f) const;
};
```

Trong đó:

- Hàm `Fraction operator+(const Fraction& f) const` trả về kết quả là tổng của phân số hiện tại với phân số **f**. Hàm này không thay đổi giá trị của phân số hiện tại.
- Hàm `Fraction operator-(const Fraction& f) const` trả về kết quả là hiệu của phân số hiện tại với phân số **f**. Hàm này không thay đổi giá trị của phân số hiện tại.
- Hàm `Fraction operator*(const Fraction& f) const` trả về kết quả là tích của phân số hiện tại với phân số **f**. Hàm này không thay đổi giá trị của phân số hiện tại.
- Hàm `Fraction operator-(const Fraction& f) const` trả về kết quả là thương của phân số hiện tại với phân số **f**. Hàm này không thay đổi giá trị của phân số hiện tại.
- Hàm `void simplify()` tối giản phân số hiện tại.
- Hàm `bool operator>(const Fraction& f) const` trả về $true$ nếu phân số hiện tại lớn hơn phân số **f**, ngược lại trả về $false$.
- Hàm `bool operator<(const Fraction& f) const` trả về $true$ nếu phân số hiện tại bé thua phân số **f**, ngược lại trả về $false$.
- Hàm `friend ostream& operator<<(ostream& os, const Fraction& a)` in ra phân số dưới dạng a/b nếu $b \neq 0$. Ngược lại in ra $invalid$. Lưu ý: cần tối giản phân số trước khi in ra kết quả.

For example:

Test	Result

`Fraction x(1, 3), y(1, 3);`
`x = x + y;`
`cout << x;`
	2/3

Answer: (penalty regime: 0 %)

```
1  class Fraction {
2    int a, b;
3  public:
4    Fraction(int a = 1, int b = 1);
5    friend ostream &operator<<(ostream &os, const Fraction &f);
6    Fraction operator+(const Fraction &f) const;
7    Fraction operator-(const Fraction &f) const;
8    Fraction operator*(const Fraction &f) const;
9    Fraction operator/(const Fraction &f) const;
10   void simplify();
11   bool operator>(const Fraction &f) const;
```

```
  bool operator<(const Fraction &f) const;
};

Fraction::Fraction(int _a, int _b) {
  a = _a;
  b = _b;
}

ostream &operator<<(ostream &os, const Fraction &f) {
  if (f.b == 0) {
    os << "invalid";
  } else {
    Fraction c = f;
    c.simplify();
    os << c.a << '/' << c.b;
  }
  return os;
}

Fraction Fraction::operator+(const Fraction &f) const {
  return Fraction(a * f.b + b * f.a, b * f.b);
}

Fraction Fraction::operator-(const Fraction &f) const {
  return Fraction(a * f.b - b * f.a, b * f.b);
}

Fraction Fraction::operator*(const Fraction &f) const {
  return Fraction(a * f.a, b * f.b);
}

Fraction Fraction::operator/(const Fraction &f) const {
  return Fraction(a * f.b, b * f.a);
}

void Fraction::simplify() {
  bool neg = (a < 0) ^ (b < 0);
  a = abs(a);
  b = abs(b);
  int g = __gcd(a, b);
  a /= g;
```

	Test	Expected	Got	

✓	`Fraction x(1, 3), y(1, 3);`
	`x = x + y;`
	`cout << x;`
		2/3	2/3	✓

✓	`Fraction x, y(-1, 2);`
	`x = x - y;`
	`cout << x;`
		3/2	3/2	✓

✓	`Fraction x(4, 5), y(5, 8);`
	`x = x * y;`
	`cout << x;`
		1/2	1/2	✓

✓	`Fraction x(3, 4), y(4, 3), z(2, 3);`
	`x = x * (y - z);`
	`cout << x;`
		1/2	1/2	✓

✓	`Fraction x(4, 5), y(5, 20), z;`
	`Fraction result = x * y / z;`
	`cout << result;`
		1/5	1/5	✓

✓	`Fraction x(1, 0);` `cout << x`	invalid	invalid	✓

	Test	Expected	Got	

✓	`Fraction x(4, 5), y(2, 5), z(-3, 5);`
	`Fraction result1 = x + y;`
	`Fraction result2 = y - z;`
	`if (result1 > result2) {`
	`  cout << "result1 > result2";`
	`} else {`
	`  cout << "result1 <= result2";`
	`}`
		`result1 > result2`	`result1 > result2`	✓

✓	`Fraction x(4, 5), y(2, 5), z(-8, 5);`
	`Fraction result1 = x + y;`
	`Fraction result2 = y - z;`
	`if (result1 < result2) {`
	`  cout << "result1 < result2";`
	`} else {`
	`  cout << "result1 >= result2";`
	`}`
		`result1 < result2`	`result1 < result2`	✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 9

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Cài đặt các hàm thành viên cho cấu trúc `Point` và lớp `Triangle` sau:

```
struct Point {
    double x, y;  // Toạ độ x, y
    Point(); // Hàm khởi tạo điểm (0, 0)
    Point(double, double); // Hàm khởi tạo từ toạ độ x, y
    Point(const Point&); // Hàm khởi tạo sao chép từ một thực thể (instance) Point khác
};

class Triangle {
    Point p1, p2, p3;
public:
    Triangle(Point, Point, Point); // Hàm khởi tạo tam giác từ 3 điểm
    double getPerimeter() const;  // Hàm trả về chu vi tam giác
    double getArea() const;  // Hàm trả về diện tích tam giác
};
```

For example:

Test	Result
Point p; cout << p.x << " " << p.y;	0 0
Point p(4, 5); cout << p.x << " " << p.y;	4 5
Point p2(4, 5); Point p(p2); cout << p.x << " " << p.y;	4 5

Point A(1,1), B(1, 4), C(5, 1);
Triangle tri(A, B, C);
cout << tri.getPerimeter() << endl;
cout << tri.getArea() << endl;
	12
	6

Answer: (penalty regime: 0 %)

```
Point::Point() {
  x = y = 0;
}

Point::Point(double _x, double _y) {
  x = _x;
  y = _y;
}

Point::Point(const Point &other) {
  *this = other;
}

double distance(const Point &p1, const Point &p2) {
  return sqrt(pow(p1.x - p2.x, 2) + pow(p1.y - p2.y, 2));
}

Triangle::Triangle(Point _p1, Point _p2, Point _p3) {
  p1 = _p1;
  p2 = _p2;
```

```
  p3 = _p3;
}

double Triangle::getPerimeter() const {
  return distance(p1, p2) + distance(p2, p3) + distance(p3, p1);
}

double Triangle::getArea() const {
  double p = getPerimeter() / 2;
  return sqrt(p * (p - distance(p1, p2)) * (p - distance(p2, p3)) * (p - distance(p3, p1)));
}
```

	Test	Expected	Got	
✓	`Point p;` `cout << p.x << " " << p.y;`	0 0	0 0	✓
✓	`Point p(4, 5);` `cout << p.x << " " << p.y;`	4 5	4 5	✓

✓	`Point p2(4, 5);`
	`Point p(p2);`
	
	`cout << p.x << " " << p.y;`
		4 5	4 5	✓

✓	`Point A(1,1), B(1, 4), C(5, 1);`
	`Triangle tri(A, B, C);`
	`cout << tri.getPerimeter() << endl;`
	`cout << tri.getArea() << endl;`
		12
		6
			12
			6
				✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi **10**

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Thiết kế lớp `Time` để biểu diễn thời gian trong ngày thoả mãn các yêu cầu sau:

- Cho phép tạo thời gian từ ba tham số nguyên **giờ**, **phút**, **giây**
- Cho phép tạo thời gian từ hai tham số nguyên **giờ**, **phút**
- Cho phép tạo thời gian từ một tham số nguyên **giờ**
- Có thể dùng lệnh $cout$ để in ra màn hình theo định dạng $hh:mm:ss$ (ví dụ **08:20:10**)
- Các biến kiểu $Time$ có thể so sánh với nhau (lớn hơn, nhỏ hơn, bằng, khác, lớn hơn hoặc bằng, nhỏ hơn hoặc bằng)
- Có hàm thành viên `int hour()` trả về giờ hiện tại
- Có hàm thành viên `int minute()` trả về phút hiện tại
- Có hàm thành viên `int second()` trả về giây hiện tại

For example:

Test	Result
`Time t(2);` `cout << t;`	02:00:00
`cout << Time(1, 10);`	01:10:00
`cout << Time(23, 0, 30);`	23:00:30
`cout << (Time(12, 4, 3) < Time(12, 3, 8));`	0
`cout << (Time(12, 4, 3) <= Time(12, 3, 8));`	0
`cout << (Time(12, 4, 3) > Time(12, 3, 8));`	1
`cout << (Time(12, 4, 3) >= Time(12, 3, 8));`	1
`cout << (Time(12, 4, 3) == Time(12, 3, 8));`	0
`cout << (Time(12, 4, 3) != Time(12, 3, 8));`	1
`Time t(13, 30, 45);` `cout << t.hour() << " " << t.minute() << " " << t.second();`	13 30 45

Answer: (penalty regime: 0 %)

```
class Time {
private:
  int h, m, s, ss;
public:
  Time(int _h = 0, int _m = 0, int _s = 0) {
    ss = _h * 3600 + _m * 60 + _s;
    h = ss / 3600;
    m = (ss % 3600) / 60;
    s = ss % 60;
  }
  friend ostream& operator<<(ostream &os, const Time &t) {
    if (t.h < 10) {
      os << '0';
    }
    os << t.h << ':';
    if (t.m < 10) {
      os << '0';
    }
    os << t.m << ':';
    if (t.s < 10) {
      os << '0';
    }
    os << t.s;
    return os;
  }
```

```
  bool operator<(Time other) { return ss < other.ss; }
  bool operator>(Time other) { return ss > other.ss; }
  bool operator<=(Time other) { return ss <= other.ss; }
  bool operator>=(Time other) { return ss >= other.ss; }
  bool operator==(Time other) { return ss == other.ss; }
  bool operator!=(Time other) { return ss != other.ss; }
  int hour() { return h; }
  int minute() { return m; }
  int second() { return s; }
};
```

	Test	Expected	Got	
✓	`Time t(2);` `cout << t;`	02:00:00	02:00:00	✓
✓	`cout << Time(1, 10);`	01:10:00	01:10:00	✓
✓	`cout << Time(23, 0, 30);`	23:00:30	23:00:30	✓
✓	`cout << (Time(12, 4, 3) < Time(12, 3, 8));`	0	0	✓
✓	`cout << (Time(12, 4, 3) <= Time(12, 3, 8));`	0	0	✓
✓	`cout << (Time(12, 4, 3) > Time(12, 3, 8));`	1	1	✓
✓	`cout << (Time(12, 4, 3) >= Time(12, 3, 8));`	1	1	✓
✓	`cout << (Time(12, 4, 3) == Time(12, 3, 8));`	0	0	✓
✓	`cout << (Time(12, 4, 3) != Time(12, 3, 8));`	1	1	✓
✓	`Time t(13, 30, 45);` `cout << t.hour() << " " << t.minute() << " " << t.second();`	13 30 45	13 30 45	✓
✓	`Time t(2, 70, 80);` `cout << t;`	03:11:20	03:11:20	✓
✓	`Time t(2, 59, 60);` `cout << t;`	03:00:00	03:00:00	✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi **11**

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Cài đặt các hàm thành viên cho lớp `BigInt` biểu diễn một số nguyên lớn như sau:

```
class BigInt {
    string value;
    int sign;
public:
    // Hàm khởi tạo số nguyên lớn
    BigInt();
    BigInt(const char*);
    BigInt(int );
    BigInt(const BigInt& );

    // Toán tử in ra màn hình
    friend ostream& operator<< (ostream& , const BigInt& );

    // Toán tử gán
    BigInt& operator=(int );
    BigInt& operator=(const char*);
    BigInt& operator=(const BigInt& );

    // Toán tử cộng
    friend BigInt operator+(const BigInt& , const BigInt & );
    friend BigInt operator+(const BigInt& , int );
    friend BigInt operator+(int , const BigInt &);

    // Toán tử trừ
    friend BigInt operator-(const BigInt& , const BigInt & );
    friend BigInt operator-(const BigInt& , int );
    friend BigInt operator-(int , const BigInt &);

    // Toán tử cộng rồi gán
    BigInt& operator+=(int );
    BigInt& operator+=(const BigInt& );

    // Toán tử trừ rồi gán
    BigInt& operator-=(int );
    BigInt& operator-=(const BigInt& );
};
```

For example:

Test	Result
`BigInt num;` `cout << num;`	`0`
`BigInt num("4879124129988949491929991249124912312");` `cout << num;`	`4879124129988949491929991249124912312`
`cout << BigInt(123);`	`123`
`BigInt num = 40;` `cout << num;`	`40`

Test	Result
BigInt num = "4441"; cout << num;	4441
BigInt tmp(23); BigInt num(tmp); cout << num;	23
BigInt a = BigInt("741239123172319239999123 9"); BigInt b = BigInt(21348123); cout << a + b << endl; cout << 43 + a << endl; cout << b + 71 << endl;	741239123172319242133936 2 741239123172319239999128 2 21348194

BigInt num("34123");
num += 23;
num += BigInt("3341");
cout << num;
	37487

Answer: (penalty regime: 0 %)

Reset answer

```
BigInt::BigInt() {
  value = "0";
  sign = 1;
}

BigInt::BigInt(const char *str) {
  string s(str);
  if (s[0] == '-') {
    sign = -1;
    s.erase(0, 1);
  } else {
    sign = 1;
    if (s[0] == '+') {
      s.erase(0, 1);
    }
  }
  value = s;
  while (value.length() > 1 && value[0] == '0') {
    value.erase(0, 1);
  }
  if (value == "0") {
    sign = 1;
  }
}

BigInt::BigInt(int num) {
  sign = num < 0 ? -1 : 1;
  value = to_string(num < 0 ? -num : num);
}

BigInt::BigInt(const BigInt &other) : value(other.value), sign(other.sign) {}

ostream &operator<<(ostream &os, const BigInt &b) {
  if (b.sign == -1 && b.value != "0")
    os << "-";
  os << b.value;
  return os;
}

BigInt &BigInt::operator=(int num) {
  *this = BigInt(num);
  return *this;
}

BigInt &BigInt::operator=(const char *str) {
  *this = BigInt(str);
  return *this;
}
```

```
BigInt &BigInt::operator=(const BigInt &other) {
  if (this != &other) {
    value = other.value;
```

	Test	Expected	Got
✓	BigInt num; cout << num;	0	0
✓	BigInt num("4879124129988949491929991249124912312"); cout << num;	4879124129988949491929991249124912312	4879124129988949491929991249124
✓	cout << BigInt(123);	123	123
✓	BigInt num = 40; cout << num;	40	40
✓	BigInt num = "4441"; cout << num;	4441	4441
✓	BigInt tmp(23); BigInt num(tmp); cout << num;	23	23
✓	BigInt a = BigInt("74123912317231924239991239"); BigInt b = BigInt(21348123); cout << a + b << endl; cout << 43 + a << endl; cout << b + 71 << endl;	74123912317231924213393627412391231723192399912821348194	74123912317231924213393627412391231723192399912821348194

✓	BigInt num("34123");
	num += 23;
	num += BigInt("3341");
	cout << num;
		37487	37487

✓	cout << BigInt(-123);	-123	-123
✓	cout << BigInt("-2312412412984712971231293817239124");	-2312412412984712971231293817239124	-2312412412984712971231293817239
✓	BigInt num = -123; cout << num;	-123	-123
✓	BigInt num = "-123"; cout << num;	-123	-123
✓	BigInt num1("-1234"); BigInt num2 = num1; cout << num2;	-1234	-1234
✓	BigInt a = "-10", b = "20", c = 5; cout << (a + b) << endl; cout << (b + a) << endl; cout << (b + c) << endl; cout << (a - b) << endl; cout << (b - a) << endl; cout << (b - c) << endl;	10 10 25 -30 30 15	10 10 25 -30 30 15
✓	BigInt a = 10; a -= 2; cout << a;	8	8

	Test	Expected	Got
✓	BigInt a = -10; a -= BigInt("4"); cout << a;	-14	-14
✓	BigInt a; a = "123249871926392163123 6123"; cout << a;	123249871926392163123 6123	123249871926392163123 6123
✓	BigInt a; a = -100; cout << a;	-100	-100

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 12

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Không dùng thư viện `string` hay `cstring`, cài đặt các hàm thành viên cho lớp `MyString` như sau:

```
class MyString {
    char *str;
    int len;
public:
    // Hàm khởi tạo xâu rỗng
    MyString();

    // Hàm khởi tạo từ một xâu kiểu c-string
    MyString(const char* );

    // Hàm khởi tạo sao chép từ một thực thể (instance) MyString khác
    MyString(const MyString& );

    // Hàm huỷ
    ~MyString();

    // Toán tử gán
    MyString operator=(const MyString& );
    MyString operator=(const char* );

    // Toán tử nối xâu rồi gán
    MyString operator+=(const MyString& );

    // Toán tử truy cập đến từng kí tự
    char& operator[] (unsigned int index);
    const char& operator[] (unsigned int index) const;

    // Hàm trả về độ dài của xâu
    int size();

    // Hàm trả về 1 thực thể MyString có các kí tự giống thực thể đang xét nhưng các chữ cái viết hoa
    MyString upper() const;

    // Hàm trả về 1 thực thể MyString có các kí tự giống thực thể đang xét nhưng các chữ cái viết thường
    MyString lower() const;

    // Toán tử để in ra màn hình
    friend ostream& operator<< (ostream& , const MyString& );

    // Toán tử nối xâu
    friend MyString operator+ (const MyString& , const MyString& );

    // Các toán tử so sánh
    friend bool operator< (const MyString& , const MyString& );
    friend bool operator> (const MyString& , const MyString& );
    friend bool operator<=(const MyString& , const MyString& );
    friend bool operator>=(const MyString& , const MyString& );
    friend bool operator==(const MyString& , const MyString& );
    friend bool operator!=(const MyString& , const MyString& );
};
```

For example:

Test	Result
cout << MyString();	

Test	Result
`cout << MyString("test");`	test
`MyString s("abc");` `cout << MyString(s);`	abc
`MyString s = "test";` `cout << s;`	test

`MyString s("test");`
`MyString s2 = s;`
`cout << s2;`
	test

`MyString s("abc");`
`s += "def";`
`cout << s;`
	abcdef

`MyString s("abcdef");` `cout << s[0] << s[3];`	ad
`const MyString s("abcdef");` `cout << s[0] << s[3];`	ad
`cout << MyString("abcd").size();`	4
`cout << MyString("AbCd").upper();`	ABCD
`cout << MyString("AbCd").lower();`	abcd

`MyString s1 = "abc", s2 = "def";`
`MyString s = s1 + s2;`
`cout << s;`
	abcdef

`cout << (MyString("abc") < MyString("abd"));`	1
`cout << (MyString("abc") <= MyString("abd"));`	1
`cout << (MyString("abc") > MyString("abd"));`	0
`cout << (MyString("abc") >= MyString("abd"));`	0
`cout << (MyString("abc") == MyString("abd"));`	0
`cout << (MyString("abc") != MyString("abd"));`	1

Answer: (penalty regime: 0 %)

```
MyString::MyString() {
  len = 0;
  str = new char[1];
  str[0] = '\0';
}

MyString::MyString(const char *s) {
  for (int i = 0; ; ++i) {
    if (s[i] == '\0') {
      len = i;
      break;
    }
  }
  str = new char[len + 1];
  for (int i = 0; i < len; ++i) {
    str[i] = s[i];
  }
  str[len] = '\0';
}

MyString::MyString(const MyString &s) {
  len = s.len;
  str = new char[len + 1];
  for (int i = 0; i < len; ++i) {
```

```
    str[i] = s[i];
  }
  str[len] = '\0';
}

MyString::~MyString() {
  delete[] str;
}

MyString MyString::operator=(const MyString &s) {
  if (this != &s) {
    *this = MyString(s);
  }
  return *this;
}

MyString MyString::operator=(const char *s) {
  *this = MyString(s);
  return *this;
}

MyString MyString::operator+=(const MyString &s) {
  char *res = new char[len + s.len + 1];
  for (int i = 0; i < len + s.len; ++i) {
    if (i < len) {
      res[i] = str[i];
    } else {
      res[i] = s.str[i - len];
```

	Test	Expected	Got	
✓	`cout << MyString();`			✓
✓	`cout << MyString("test");`	test	test	✓
✓	`MyString s("abc");` `cout << MyString(s);`	abc	abc	✓
✓	`MyString s = "test";` `cout << s;`	test	test	✓

✓	`MyString s("test");`
	`MyString s2 = s;`
	`cout << s2;`
		test	test	✓

✓	`MyString s("abc");`
	`s += "def";`
	`cout << s;`
		abcdef	abcdef	✓

✓	`MyString s("abcdef");` `cout << s[0] << s[3];`	ad	ad	✓
✓	`const MyString s("abcdef");` `cout << s[0] << s[3];`	ad	ad	✓
✓	`cout << MyString("abcd").size();`	4	4	✓
✓	`cout << MyString("AbCd").upper();`	ABCD	ABCD	✓
✓	`cout << MyString("AbCd").lower();`	abcd	abcd	✓

✓	`MyString s1 = "abc", s2 = "def";`
	`MyString s = s1 + s2;`
	`cout << s;`
		abcdef	abcdef	✓

✓	`cout << (MyString("abc") < MyString("abd"));`	1	1	✓
✓	`cout << (MyString("abc") <= MyString("abd"));`	1	1	✓
✓	`cout << (MyString("abc") > MyString("abd"));`	0	0	✓

	Test	Expected	Got	
✓	`cout << (MyString("abc") >= MyString("abd"));`	`0`	`0`	✓
✓	`cout << (MyString("abc") == MyString("abd"));`	`0`	`0`	✓
✓	`cout << (MyString("abc") != MyString("abd"));`	`1`	`1`	✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Trở lại Khoá học