**Exercitiul 11**

////////////////////////////////////

//// 1

////////////////////////////////////

#include <iostream>

using namespace std;

class Vector2D {

int x, y;

public:

Vector2D(const int& a, const int& b) : x(a), y(b) {}

Vector2D operator+ (Vector2D v) {

Vector2D w(x + v.x, y + v.y); return w;

}

friend ostream& operator<< (ostream& out, Vector2D& v) {

out << "(" << v.x << ", " << v.y << ")"; return out;

}

};

int main () {

cout << Vector2D(10, 5) + Vector2D(22, 159);

return 0;

}

// Secventa nu compileaza din cauza liniei 16 deoarece operatorul << poate fi apelat doar cu obiected de tip Vector2D care au adresa in memorie (i.e. non constante)

// Rezolvare:

// 1 - modificare linia 12 in: friend ostream& operator<< (ostream& out, const Vector2D& v) {

// 2 - modificare linia 12 in: friend ostream& operator<< (ostream& out, Vector2D v) {

// Barem:

// Total: 0.5p

// 1.a 0.1p secventa nu compileaza cu explicatii care nu sunt corecte dar in zona supraincari de operator

// 1.b 0.25p secventa nu compileaza cu identificare corecta a problemei conform explicatiei de mai sus

// 2 0.25p rezolvarea problemei prin una din solutiile indicate

// OBS! Doar una dintre 1.a sau 1.b poate fi alocata.

// OBS! Punctajul de punctul 2 se aloca doar daca eroare la compilare este identificata corect.

////////////////////////////////////

//// 2

////////////////////////////////////

#include <iostream>

using namespace std;

class Vector2D {

int x, y;

public:

Vector2D(const int& a, const int& b) : x(a), y(b) {}

Vector2D operator+ (Vector2D v) {

Vector2D w(x + v.x, y + v.y); return w;

}

friend ostream& operator<< (ostream& out, const Vector2D& v) {

out << "(" << v.x << ", " << v.y << ")"; return out;

}

};

int main () {

cout << Vector2D(2, 5) + Vector2D(4, 8);

return 0;

}

// Secventa compileaza si afiseaza (6, 13)

// Barem:

// Total: 0.5p

// 1.a 0.1p secventa compileaza si afiseaza (x, y) - i.e orice 2 numere intregi

// 1.b 0.25p secventa compileaza si afiseaza (6, 13)

// 2 0.25p explicare comportamentului secventei de cod (daca explicatiile sunt partial corecte, ~50%, se acorda jumatate din punctaj)

// OBS! Doar una dintre 1.a sau 1.b poate fi alocata.

////////////////////////////////////

//// 3

////////////////////////////////////

#include <iostream>

using namespace std;

class Vector2D {

int x, y;

public:

Vector2D(const int& a, const int& b) : x(a), y(b) {}

friend Vector2D operator+ (Vector2D& v, Vector2D& u) {

Vector2D w(u.x + v.x, u.y + v.y); return w;

}

friend ostream& operator<< (ostream& out, const Vector2D& v) {

out << "(" << v.x << ", " << v.y << ")"; return out;

}

};

int main () {

cout << Vector2D(45, 29) + Vector2D(87, 10);

return 0;

}

// Secventa nu compileaza din cauza liniei 16 deoarece operatorul + se asteapta la 2 obiecte Vector2D cu adresa in memorie, iar obiectele anonime sunt obiecte constante

// Rezolvare

// 1. modificarea liniei 8 in: friend Vector2D operator+ (const Vector2D& v, const Vector2D& u) {

// 2. modificarea liniei 8 in: friend Vector2D operator+ (Vector2D v, Vector2D u) {

// Barem:

// Total: 0.5p

// 1.a 0.1p secventa nu compileaza cu explicatii care nu sunt corecte

// 1.b 0.25p secventa nu compileaza cu explicatii corecte

// 2 0.25p Rezolvarea problemei de compilare prin solutia indicata

// OBS! Doar una dintre 1.a sau 1.b poate fi alocata.

// OBS! Punctajul de punctul 2 se aloca doar daca eroare la compilare este identificata corect.

////////////////////////////////////

//// 4

////////////////////////////////////

#include <iostream>

using namespace std;

class Vector2D {

int x, y;

const int translate = 10 ;

public:

Vector2D(const int& a, const int& b) : x(a), y(b) {}

Vector2D operator\* (const Vector2D& v) {

return Vector2D(translate + x \* v.x, translate + y + v.y);

}

friend ostream& operator<< (ostream& out, const Vector2D& v) {

out << "(" << v.x << ", " << v.y << ")"; return out;

}

};

int main () {

cout << Vector2D(98, 69) \* Vector2D(82, 12);

return 0;

}

// Secventa compileaza si afiseaza (8046, 91)

// Barem:

// Total: 0.5p

// 1.a 0.1p secventa compileaza si afiseaza (x, y) - i.e orice 2 numere intregi

// 1.b 0.25p secventa compileaza si afiseaza (8046, 91)

// 2 0.25p explicare comportamentului secventei de cod (daca explicatiile sunt partial corecte, ~50%, se acorda jumatate din punctaj)

// OBS! Doar una dintre 1.a sau 1.b poate fi alocata.

////////////////////////////////////

//// 5

////////////////////////////////////

#include <iostream>

using namespace std;

class Vector2D {

int x, y;

public:

Vector2D(const int& a, const int& b) : x(a), y(b) {}

Vector2D operator\* (Vector2D v) const {

return Vector2D(x + v.y, y + v.x);

}

friend ostream& operator<< (ostream& out, const Vector2D& v) {

out << "(" << v.x << ", " << v.y << ")"; return out;

}

};

int main () {

cout << Vector2D(33, 40).operator\*(Vector2D(77, 60));

return 0;

}

// Secventa compileaza si afiseaza (93, 117)

// Barem:

// Total: 0.5p

// 1.a 0.1p secventa compileaza si afiseaza (x, y) - i.e orice 2 numere intregi

// 1.b 0.25p secventa compileaza si afiseaza (93, 117)

// 2 0.25p explicare comportamentului secventei de cod (daca explicatiile sunt partial corecte, ~50%, se acorda jumatate din punctaj)

// OBS! Doar una dintre 1.a sau 1.b poate fi alocata.