**Exercitiul 10**

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

//// 1

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

#include <iostream>

using namespace std;

class C {

int const \*p;

public:

C (int \*q) : p(q) {}

void reload () { delete p; p = new int;}

void set (const int \* const q) { \*p = \*q; }

};

int main () {

int x = 20210614;

C ob(&x);

const int& rx = x;

ob.reload(); ob.set(&rx);

return 0;

}

// Secventa nu compileaza din cauza liniei 9 deoarece p este un pointer catre constanta (isi poate modifica adresa, nu poate modifica valoarea salvata la aceea adresa)

// Rezolvare:

// modificarea liniei 5 in: int \*p;

// 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.

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

//// 2

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

#include <iostream>

using namespace std;

class C {

int \*p;

public:

C (int \*q) : p(q) {}

void reload () { delete p; p = new int;}

void set (int q) const { \*p = q; }

};

int main () {

int x = 20210614;

C ob(&x);

ob.reload(); ob.set(x);

return 0;

}

// Secventa compileaza si afiseaze eroare la runtime deoarece se incearca dezalocarea unei adrese care nu a fost alocata dinamic:

// main(57571,0x11b544e00) malloc: \*\*\* error for object 0x7ffee2ab15b8: pointer being freed was not allocated

// main(57571,0x11b544e00) malloc: \*\*\* set a breakpoint in malloc\_error\_break to debug

// Abort trap: 6

// Barem:

// Total: 0.5p

// 1.a 0.1p secventa compileaza si afiseaza orice mai putin raspunsul corect

// 1.b 0.25p secventa copileaza si afiseaza exceptia la dezalocare

// 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 C {

int \* p;

public:

C (int \*q) : p(q) {}

void reload () { delete p; p = new int;}

void set (const int \* const q) { \*p = \*q + 13; }

operator int () {return \*p;}

};

int main () {

int \*x = new int(20210601); const int& rx = \*x;

C ob(x);

ob.reload(); ob.set(&rx);

cout << ob;

return 0;

}

// Secventa compileaza si afiseaze 20210614

// Barem:

// Total: 0.5p

// 1.a 0.1p secventa compileaza si afiseaza un numar apropiat de 20210614, dar nu 20210614

// 1.b 0.25p secventa compileaza si afiseaza 20210614

// 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.

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

//// 4

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

#include <iostream>

using namespace std;

class C {

int \* const p;

public:

C (int q) : p(new int(q)) {}

void set (const int& q) const { \*p = q + 86; }

operator int () const {return \*p;}

};

int main () {

const C ob(91973549);

ob.set(422032);

cout << ob;

return 0;

}

// Secventa compileaza si afiseaze 422118

// Barem:

// Total: 0.5p

// 1.a 0.1p secventa compileaza si afiseaza un numar apropiat de 422118, dar nu 422118

// 1.b 0.25p secventa compileaza si afiseaza 422118

// 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 C {

int \* const p;

public:

C (int q) : p(new int(q)) {}

void set (const int& q) const { \*p = q + 59; }

operator int () {return \*p;}

};

int main () {

const C ob(22973890);

ob.set(488474);

cout << ob;

return 0;

}

// Secventa nu compileaza din cauzaa liniei 14, deoarece operatorul << nu este supraincarcat pt clasa C, iar operatorul de conversie la int nu poate fi folosit deoarece obiectul ob este constant

// Rezolvare:

// modificare liniei 9 in: operator int () const {return \*p;}

// 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.