```
// Exercitii extrase de pe GeeksforGeeks
/*
* 1. Spuneti daca programul de mai jos este corect. In caz afirmativ,
* spuneti ce afisaza, in caz negativ spuneti ce nu este corect.
*/
/*

#include <iostream>
using namespace std;
class A
{
public:
   void print() { cout << "A::print()"; }
};
class B : private A
{
public:
   void print() { cout << "B::print()"; }
};
class C : public B
{
public:
   void print() { A::print(); }
};
int main()
{
   C b;
   b.print();
}
Raspuns: Programul va da eroare de compilare deoarece B mosteneste A in mod privat
si chiar daca C il mosteneste pe B in mod public nu vom putea avea acces la metoda print din A
deoarece aceasta devine metoda privata in clasa B, deci nu va putea fi accesata din C
O posibila rezolvare:
Clasa B il mosteneste pe A in mod public sau protected, oricare dintre aceste variante ar fi corecta
pentru examen.
*/
/*
#include <iostream>
using namespace std;
class A
{
public:
   void print() { cout << "A::print()"; }
};
class B : public A
{
public:
   void print() { cout << "B::print()"; }
};
class C : public B
{
public:
```

```
    void print() { A::print(); }
};
int main()
{
    C b;
    b.print();
}
*/

/*
* 2. Spuneti daca programul de mai jos este corect. In caz afirmativ,
* spuneti ce afisaza, in caz negativ spuneti ce nu este corect.
*/
/*
#include<iostream>
using namespace std;
class base {
public:
    void show() { cout << " In Base \n"; }
};

class derived: public base
{
    int x;
public:
    void show() { cout<<"In derived \n"; }
    derived()
    { x = 10; }
    int getX() const { return x;}
};

int main()
{
    derived d;
    base *bp = &d;
    bp->show();
    cout << bp->getX();
    return 0;
}
```

Raspuns: Programul va da eroare de compilare la instructiunea: cout << bp->getX() deoarece clasa de baza nu contine
metoda int getX() const { return x; }. In acest program se realizeaza procedeul de Upcasting : mai intai este declarat
si instantiat pe stiva un obiect de tip "derived", apoi se face Upcasting printr-un pointer de tipul clasei "base" care
va face referire la obiectul anterior. Instructiunea bp->show() ar rula si ar afisa " In base \n" dar cand compilatorul
ajunge la linia in care se incearca accesarea metodei bp->getX() acesta va semnala ca clasa base nu are un asemenea membru.
Puteti avea in minte mereu schema pe care Bogdan va desenat-o pe tabla la tutoriat si anume ganditi-va la clase ca

la niste cutii cum e si aicea clasa "base" reprezinta o cutie mai mica, iar clasa "derived" o cutie care o va contine
pe cea "base", dar va puteti imagina ca aveti acces doar la cutia mai mica deci nu ati putea chestiile din cutia mai mare.
*/
/*
O posibila rezolvare: Schimbam tipul pointer-ului bp si il facem de tip "derived"
#include<iostream>
using namespace std;
class base {
public:
   void show() { cout << " In Base \n"; }
};

class derived: public base
{
   int x;
public:
   void show() { cout<<"In derived \n"; }
   derived()
   { x = 10; }
   int getX() const { return x;}
};

int main()
{
   derived d;
   derived *bp = &d;
   bp->show();
   cout << bp->getX();
   return 0;
}
*/

/*
* 3. Spuneti daca programul de mai jos este corect. In caz afirmativ,
* spuneti ce afisaza, in caz negativ spuneti ce nu este corect.
*/
/*
#include <iostream>
using namespace std;
class B
{
   int x;
public:
   B(int v) {v=x;}
   int get_x() {return x;}
};

class D: private B
{
   int y;

```
public:
   D(int v): B(v) {}
   int get_x() {return x;}
};

int main(){
   D d(10);
   cout<<d.get_x();
   return 0;
}
```

Raspuns: Programul nu va trece de compilare deoarece clasa "D" mosteneste clasa "B", iar cum x este un
membru privat al clasei B acesta nu va fi "mostenit" in clasa D deci prin urmare nu va putea fi accesat.
*/
/*
O posibila rezolvare: Il facem pe x membru protected sau public(nerecomandat). Atentie mare aici rezultatul
rularii va fi unul random desi in unele cazuri are compilatorul grija sa faca zonele de memorie 0 acest
lucru nu este garantat.

```
#include <iostream>
using namespace std;
class B
{
protected:
   int x;
public:
   B(int v) {v=x;}
   int get_x() {return x;}
};

class D: private B
{
   int y;
public:
   D(int v): B(v) {}
   int get_x() {return x;}
};

int main(){
   D d(10);
   cout<<d.get_x();
   return 0;
}
```

*/

```
//Exercitii extrase din examene
/*
 * 4. Spuneti daca programul de mai jos este corect. In caz afirmativ,
 * spuneti ce afisaza, in caz negativ spuneti ce nu este corect.
 */
/*
#include <iostream>

using namespace std;
class B
{   int a;
    B(int i=0) { a=i; }
    int get_a() { return a; } };
class D: protected B
{ public: D(int x=0): B(x) {}
    int get_a() { return B::get_a(); } };
int main()
{ D d(-89);
    cout<<d.get_a();
    return 0;
}
Raspuns: Programul nu va trece de compilare deoarece toti membri clasei B sunt declarati ca si
private inclusiv
constructorul acestuia este redefinit, deci nu vom putea instantia un obiect folosind pattern-ul clasei
B. Astfel
cand se incearca crearea unui obiect de tipul clasei D compilatorul va semnala o eroare ca
constructorul clasei B
este private.
*/
/*
O posibila solutie: Facem constructorul si metoda get_a() membri public sau protected.
*/
/*
#include <iostream>
using namespace std;
class B
{   int a;
 public:
    B(int i=0) { a=i; }
    int get_a() { return a; } };
class D: protected B
{ public: D(int x=0): B(x) {}
    int get_a() { return B::get_a(); } };
int main()
{ D d(-89);
    cout<<d.get_a();
    return 0;
}
*/

/*
 * 5. Spuneti daca programul de mai jos este corect. In caz afirmativ,
```

```
 * spuneti ce afisaza, in caz negativ spuneti ce nu este corect.
 */
/*
#include <iostream>
using namespace std;

class B {
   protected:
      int x;
   public:
      B(int i = 16) { x = i; }
      B f(B ob) { return x + ob.x; }
      void afisare() { cout << x; }
};

class D: public B {
   public:
      B f(B ob) {
         return x + 1;
      }
};

int main() {
   B *p1 = new D, *p2 = new B, *p3 = new B(p1->f(*p2));
   p3->afisare();
   return 0;
}
```

Raspuns: Programul trece de compilare cu succes si va afisa 32. Va recomand sa luati programele de genul acesta
pe foaie si sa le rulati pas cu pas exact cum ar face compilatorul ca sa va prindeti de raspuns.
In cazul de fata pointer-ul p1 va face referire catre un obiect de tipul clasei D, deci membrul x al sau va fi initializat cu valoarea 16 deoarece este apelat constructorul implicit al clasei D care va apela la randul lui constructorul cu parametru implicit al clasei B in cazul nostru (i = 16). Apoi pointer-ul p2
va face referire la un obiect de tipul clasei B deci de asemenea i se va apela constructorul cu valoare implicita
deci x-ul lui va fi de asemenea 16. Pointer-ul p3 va fi instantiant folosind clasa B dar folosind o metoda
definita din p1 si anume f care primeste ca parametru obiectul p2. Acum aceasta functie va intoarce un obiect care
are in membrul x valoarea 32 deoarece x-ul lui p1 este 16 si cel al lui p2 este de asemenea 16. Stiu ca pare ciudat,
dar asa functioneaza lucrurile in C++ pentru ca functia intoarce un obiect de tip B, dar acest obiect are in constructor
un intreg de tip int si atunci compilatorul stie singur sa creeze un obiect e ca si cum ar apela singur constructorul cu
parametru 32. In realitate de multe ori in cazuri ca acesta compilatorul stie sa faca ceva ce se numeste Copy Ellision.

```
 */


/*
```

```
 * 6. Spuneti daca programul de mai jos este corect. In caz afirmativ,
 * spuneti ce afisaza, in caz negativ spuneti ce nu este corect.
 */

/*
#include <iostream>
using namespace std;

class B {
   public:
      int x;
      B(int i = 16) { x = i; }
      B f(B ob) { return x + ob.x; }
};

class D: public B {
   public:
      D(int i = 25) {
         x = i;
      }

      B f(B ob) {
         return x + ob.x + 1;
      }

      void afisare() {
         cout << x;
      }
};

int main() {
   B *p1 = new D, *p2 = new B, *p3 = new B(p1->f(*p2));
   cout << p3 -> x;
   return 0;
}
Raspuns: 41. Programul este asemanator cu cel de la exercitiul 5. Incercati sa il faceti singuri.
*/

/*
 * 7. Spuneti daca programul de mai jos este corect. In caz afirmativ,
 * spuneti ce afisaza, in caz negativ spuneti ce nu este corect.
 */

/*
#include <iostream>
using namespace std;

class A {
   public:
      int x;
      A(int i = 0) {
         x = i;
```

```
        }
        A minus() {
            return 1-x;
        }
};

class B : public A{
    public: B(int i = 0) { x = i; }
            void afisare() { cout << x; }
};

int main() {
    A *p1 = new B(18);
    *p1 = p1->minus();
    p1->afisare();
    return 0;
}
```
Raspuns: Programul nu va trece de compilare deoarece clasa A nu contine metoda afisare. Chiar daca
noi instantiem un obiect de tipul B in main se realizeaza procedeul de upcasting deci vom face referire
la el printr-un obiect de tipul clasei A care nu are definita metoda afisare().
*/
/*
O posibila solutie: Definim metoda afisare() si in clasa A.
```
#include <iostream>
using namespace std;

class A {
    public:
        int x;
        A(int i = 0) {
            x = i;
        }
        A minus() {
            return 1-x;
        }
        void afisare() { cout << x; }
};

class B : public A{
    public: B(int i = 0) { x = i; }
            void afisare() { cout << x; }
};

int main() {
    A *p1 = new B(18);
    *p1 = p1->minus();
    p1->afisare();
    return 0;
}
```
*/

```
/*
 * 8. Spuneti daca programul de mai jos este corect. In caz afirmativ,
 * spuneti ce afisaza, in caz negativ spuneti ce nu este corect.
 */
/*
#include <iostream>
using namespace std;

class A {
protected: int x;
public: A(int i = -16) { x = i; }
        A f(A a) { return x + a.x; }
        void afisare() { cout << x; }
};

class B: public A {
public: B(int i = 3): A(i) {}
        B f(B b) { return x + b.x + 1; }
};

int main() {
    A *p1 = new B, *p2 = new A, *p3 = new A(p1->f(*p2));
    p3->afisare();
    return 0;
}
Raspuns: -13. Seamana cu exercitiile de mai sus incercati sa o faceti singuri.
*/

/*
 * 9. Spuneti daca programul de mai jos este corect. In caz afirmativ,
 * spuneti ce afisaza, in caz negativ spuneti ce nu este corect.
 */
/*
#include <iostream>
using namespace std;

class A {
public: int x;
        A(int i = 0) { x = i; }
        A minus() { return 1 - x; }
};

class B: public A {
public: B(int i = 0) { x = i; }
        void afisare() { cout << x; }
};

int main() {
    A *p1 = new B(18);
    *p1 = p1->minus();
    dynamic_cast<A*>(p1)->afisare();
```

```
    return 0;
}
```
Raspuns: Programul nu va trece de compilare. Motivul este acelasi ca la problema 7. Dynamic_cast-ul acesta
nu are nici un farmec aici e pus doar asa de forma mai mult.
O posibila solutie: La fel ca si la problema 7
*/

```
/*
 * 10. Spuneti daca programul de mai jos este corect. In caz afirmativ,
 * spuneti ce afisaza, in caz negativ spuneti ce nu este corect.
 */
/*
#include<iostream>
using namespace std;
class B
{ int i;
public: B() { i=1; }
int get_i() { return i; }
};
class D: public B
{ int j;
public: D() { j=2; }
    int get_i() {return B::get_i()+j; }
};
int main()
{ const int i = cin.get();
    if (i%3) { D o;}
    else {B o;}
    cout<<o.get_i();
    return 0;}
```
Raspuns: Programul nu va trece de compilare deoarece obiectul "o" este creat indiferent de caz in if sau else
deci nu va fi accesibil in blocul exterior acestora si cand se incearca accesarea lui compilatorul
va semnala o eroare ca obiectul "o" nu a fost declarat in scopul blocului respectiv(cel al lui main).
*/
/*
O posibila rezolvare: Ca sa treaca de compilare putem declara un pointer de tipul clasei B si sa instantiem folosind
procedeul de Upcasting.
```
#include<iostream>
using namespace std;
class B
{ int i;
public: B() { i=1; }
    int get_i() { return i; }
};
class D: public B
{ int j;
public: D() { j=2; }
    int get_i() {return B::get_i()+j; }
```

```
};
int main()
{ const int i = cin.get();
   B *o;
   if (i%3) {o = new D;}
   else { o = new B;}
   cout<<o->get_i();
   return 0;}
*/

/*
 * 11. Spuneti daca programul de mai jos este corect. In caz afirmativ,
 * spuneti ce afisaza, in caz negativ spuneti ce nu este corect.
 */
/*
#include <iostream>
#include <typeinfo>

using namespace std;
class B
{ int i;
public: B() { i=1; }
   int get_i() { return i; }
};
class D: B
{ int j;
public: D() { j=2; }
   int get_j() {return j; }
};
int main()
{ B *p=new D;
   cout<<p->get_i();
   if (typeid((B*)p).name()=="D*") cout<<((D*)p)->get_j();
   return 0;
}
```

Raspuns: Programul nu va trece de compilare, deoarece se incearca realizarea procedurii de Upcasting dar acest lucru
nu este posibil decat daca mostenirea se realizeaza in mod public, altfel compilatorul va semnala o eroare. (A se tine minte).
*/
/*
O posibila solutie: Il mostenim in mod public pe B in clasa D.

```
#include <iostream>
#include <typeinfo>

using namespace std;
class B
{ int i;
public: B() { i=1; }
   int get_i() { return i; }
};
class D: public B
```

```
{ int j;
public: D() { j=2; }
   int get_j() {return j; }
};
int main()
{ B *p=new B;
   cout<<p->get_i();
   if (typeid((B*)p).name()=="D*") cout<<((D*)p)->get_j();
   return 0;
}
*/


/*
 * 12. Spuneti daca programul de mai jos este corect. In caz afirmativ,
 * spuneti ce afisaza, in caz negativ spuneti ce nu este corect.
 */
/*
#include<iostream>
using namespace std;
class B
{ protected: int x;
public: B(int i=28) { x=i; }
   B f(B ob) { return x+ob.x+1; }
   void afisare(){ cout<<x; } };
class D: public B
{ public: D(int i=-32):B(i) {}
   B f(B ob) { return x+ob.x-1; } };
int main()
{ B *p1=new D, *p2=new B, *p3=new B(p1->f(*p2));
   p3->afisare();
   return 0;
}
Raspuns: Programul nu va trece de compilare deoarece se incearca accesarea unui membru
protected in clasa D( si anume x).
*/
/*
O solutie posibila: Facem membrul x al clasei B public
#include<iostream>
using namespace std;
class B
{ public: int x;
public: B(int i=28) { x=i; }
   B f(B ob) { return x+ob.x+1; }
   void afisare(){ cout<<x; } };
class D: public B
{ public: D(int i=-32):B(i) {}
   B f(B ob) { return x+ob.x-1; } };
int main()
{ B *p1=new D, *p2=new B, *p3=new B(p1->f(*p2));
   p3->afisare();
   return 0;
```

```
}
*/
```