Exercițiul 1

Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect și realizați o modificare astfel încât acesta să compileze fără a-i schimba funcționalitatea.

```
#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();
}
```

Exercițiul 2

Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect și realizați o modificare astfel încât acesta să compileze fără a-i schimba funcționalitatea.

```
#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;
}
```

Exercițiul 3

Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect și realizați o modificare astfel încât acesta să compileze fără a-i schimba funcționalitatea.

```
#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;
}
```

Exercițiul 4

Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect și realizați o modificare astfel încât acesta să compileze fără a-i schimba

funcționalitatea.

```
#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;
}
```

Exercițiul 5

Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect și realizați o modificare astfel încât acesta să compileze fără a-i schimba funcționalitatea.

```
#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;
}
```

Exercițiul 6

Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect și realizați o modificare astfel încât acesta să compileze fără a-i schimba funcționalitatea.

```
#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;
}
```

Exercițiul 7

Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect și realizați o modificare astfel încât acesta să compileze fără a-i schimba

funcționalitatea.

```
#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;
}
```

Exercițiul 8

Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect și realizați o modificare astfel încât acesta să compileze fără a-i schimba funcționalitatea.

```
#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;
}
```

Exercițiul 9

Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect și realizați o modificare astfel încât acesta să compileze fără a-i schimba funcționalitatea.

```
#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;
}
```

Exercițiul 10

Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect și realizați o modificare astfel încât acesta să compileze fără a-i schimba funcționalitatea.

```
#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;
}
```

Exercițiul 11

Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect și realizați o modificare astfel încât acesta să compileze fără a-i schimba funcționalitatea.

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

class Base
{
public:
    int fun()      { cout << "Base::fun() called"; }
    int fun(int i) { cout << "Base::fun(int i) called"; }
};

class Derived: public Base
{
public:
```

```
    int fun(char x)   { cout << "Derived::fun(char ) called"; }
};

int main()
{
    Derived d;
    d.fun();
    return 0;
}
```