```
//set 2
#include<iostream>

using namespace std;
class Base1 {
  public:
      Base1()
      { cout << " Base1's constructor called"<<endl;    }
 };

 class Base2 {
  public:
      Base2()
      { cout << "Base2's constructor called"<<endl;    }
 };

 class Derived: public Base1, public Base2 {
    public:
      Derived()
      {  cout << "Derived's constructor called"<<endl;    }
 };

 int main()
 {
    Derived d;
    return 0;
 }




 //-------------------------------------------------------------------------------//




//set 3
#include<iostream>
using namespace std;

 class base {
     int arr[10];
 };

class b1: public base { };

class b2: public base { };

 class derived: public b1, public b2 {};

 int main(void)
 {
   cout<<sizeof(derived);
   getchar();
   return 0;
 }
```

```
//--------------------------------------------------------------------------------//


//set4
#include<iostream>

using namespace std;
class P {
 public:
    void print()
    { cout <<" Inside P::"; }
 };

class Q : public P
{ public:
    void print()
    { cout <<" Inside Q"; }
 };

 class R: public Q {
 };

 int main(void)
 {
   R r;

   r.print();
   return 0;
 }



 //--------------------------------------------------------------------------------//


 //set 15
#include <iostream>
using namespace std;

 classA
 {
 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();
 }
```

```
//set 16
#include<iostream>
using namespace std;

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

 class Derived: public Base
 {
 public:
     intfun(charx)     { cout << "Derived::fun(char ) called";}
 };

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

```
 //-------------------------------------------------------------------------------//
```

```
//set 16
#include<iostream>
using namespace std;
class Base
 {
    protected:
       int x;
    public:
       Base (int i){ x = i;}
 };

 class Derived : public Base
 {
    public:
       Derived (int i):x(i) { }
       void print() { cout << x ;}
 };

 int main()
 {
     Derived d(10);
     d.print();
 }
```

```
/* Exercitiul 1
#include<iostream>

using namespace std;

class B1 {public: int x;};
class B2 {public: int y;};
class B3 {public: int z;};
class B4 {public: int t;};
class D: public B1, private B2, protected B3, B4 {public: int u;};
int main(){
    D d;

    cout<<d.u;
    cout<<d.x;
    cout<<d.y;
    cout<<d.z;
    cout<<d.t;
    return 0;
}
*/
/*
 * Dar daca in clasele "B1", "B2", "B3", "B4";
 * in loc de "public" scriam "private" sau "protected"
 */

/*
 * Exercitiul 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 B
{
protected:
    int a;
public:
    B() { a=7; }
};

class D: public B
{
public:
    int b;
    D() { b=a+7; }
};

int main() {
    D d;
    cout<< d.b;
    return 0;
}
*/
```

```
/*
 * Exercitiul 3
 * Spuneţi de câte ori se execută fiecare constructor în programul de mai jos şi în
 * ce ordine.
 */
/*
#include <iostream>
using namespace std;
class cls1
{ protected: int x;
public: cls1(){ x = 13; } };
class cls2: public cls1
{ int y;
public: cls2(){ y = 15; }
    int f(cls2 ob) { return (ob.x+ob.y); } };
int main()
{ cls2 ob; cout<<ob.f(ob);
    return 0;
}
*/

/*
 * Exercitiul 4
 * Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce
 * afişează, altfel, spuneţi de ce nu este corect.
 */
/*
#include <iostream>
using namespace std;
class cls1
{ protected: int x;
public: cls1(int i=10) { x=i; }
    int get_x() { return x;} };
class cls2: cls1
{ public: cls2(int i):cls1(i) {} };
int main()
{ cls2 d(37);
    cout<<d.get_x();
    return 0;
}
*/
```

```
/*
 * Exercitiul 5
 * Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce
 * afişează, altfel, spuneți de ce nu este corect.
 */
/*
#include <iostream>
using namespace std;
class B1 { public: int x; };
class B2 { int y; };
class B3 { public: int z; };
class B4 { public: int t; };
class D: private B1, protected B2, public B3, B4
{ int u; };
int main()
{ D d;
    cout<<d.u;
    cout<<d.x;
    cout<<d.y;
    cout<<d.z;
    cout<<d.t;
    return 0;
}
*/

/*
 * Exercitiul 6
 * Spuneți de câte ori se apelează fiecare constructor
 * în programul de mai jos şi în ce ordine.
 */
/*
#include <iostream>
using namespace std;
class cls1
{ protected: int x;
public: cls1(){ x=13; } };
class cls2: public cls1
{ protected: int y;
public: cls2(){ y=15; } };
class cls3: public cls2
{ protected: int z;
public: cls3(){ z=17; }
    int f(cls3 ob){ return ob.x+ob.y+ob.z; } };

int main()
{ cls3 ob;
    ob.f(ob);
    return 0;
}
*/
```

```
/*
 * Exercitiul 7
 * Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afişează,
în caz
 * negativ spuneți de ce nu este corect.
 */
/*
#include <iostream>
using namespace std;
class B
{ int x;
public: B(int i=0) { x = i; } };

class D: public B
{ public: D():B(15) {}

    int f() { return x; } };

int main()
{   D d;
    cout<<d.f();
    return 0;
}
*/
```

```
/*
 * Exercitiul 8
 * Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează,
în caz
 * negativ spuneți de ce nu este corect.
 */
/*
#include <iostream>
using  namespace std;
class B {
    static int x;
    int i;
    public: B() {
        x++;
        i = 1;
    }
        ~B() {
            x--;
        }

        static int get_x() { return x; }
        int get_i() { return i; }
};
int B::x;
class D: public B {
public: D() { x++; }
        ~D() { x--; }
};

int f(B *q) {
    return (q->get_i())  + 1;
}

int main() {
    B *p = new B;
    cout << f(p);
    delete p;
    p = new D;
    cout << f(p);
    delete p;
    cout << D::get_x();
    return 0;
}
*/
```

```
/*
 * Exercitiul 9
 * Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afişează,
în caz
 * negativ spuneți de ce nu este corect.
 */

/*
#include <iostream>
using namespace std;

class A {
    protected:
        int x;
    public:
        A(int i = 14) { x = i; }
};

class B: A {
    public:
        B(B &b) {
            x = b.x;
        }
        void afisare() {
            cout << x;
        }
};

int main() {
    B b1, b2(b1);
    b2.afisare();
    return 0;
}
*/

/*
 * Exercitiul 10
 * Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afişează,
în caz
 * negativ spuneți de ce nu este corect.
 */
/*
#include <iostream>
using namespace std;
class A
{ protected: int x;
public: A(int i=14) { x=i; } };
class B: A
{ public: B():A(2){}
    B(B& b) { x=b.x-14; }
    void afisare() { cout<<x; } };
int main()
{ B b1, b2(b1);
    b2.afisare();
    return 0;
}
*/
```

```
/*
 * Exercitiul 11
 * Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează,
în caz
 * negativ spuneți de ce nu este corect.
 */
/*
#include<iostream>
using namespace std;
class B
{ protected: static int x;
    int i;
public: B() { x++; i=1; }
    ~B() { x--; }
    static int get_x() { return x; }
    int get_i() { return i; } };
int B::x;
class D: public B
{ public: D() { x++; i++;}
    ~D() { x--; i--;}
    int f1(B o){return 5+get_i();} };
int f(B *q)
{ return (q->get_x())+1; }
int main()
{ B *p=new B[10];
    cout<<f(p);
    delete[] p;
    p=new D;
    cout<< p->f1(p);
    delete p;
    cout<<D::get_x();
    return 0;
}
*/
/*
 * Exercitiul 12
 * Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează,
în caz
 * negativ spuneți de ce nu este corect.
 */
/*
#include<iostream>
using namespace std;

class A
{ int x;
public: A(int i):x(i){}
    int get_x(){ return x; } };
class B: public A
{ int y;
public: B(int i,int j):y(i),A(j){}
    int get_y(){ return y; } };
class C: protected B
{ int z;
public: C(int i,int j,int k):z(i),B(j,k){}
    int get_z(){ return z; } };
int main() {
    C c(1, 2, 3);
    cout << c.get_x() + c.get_y() + c.get_z();
    return 0;
}
*/
```