Problem 1.a

```
#include<iostream>
using namespace std;
class A{
    int a;
    void modify_a(int x){
        a = x;
    }
};
int main(){
    A a;
    a.modify_a(10);
    cout<<a.a<<endl;
    return 0;
}
```

Output ?

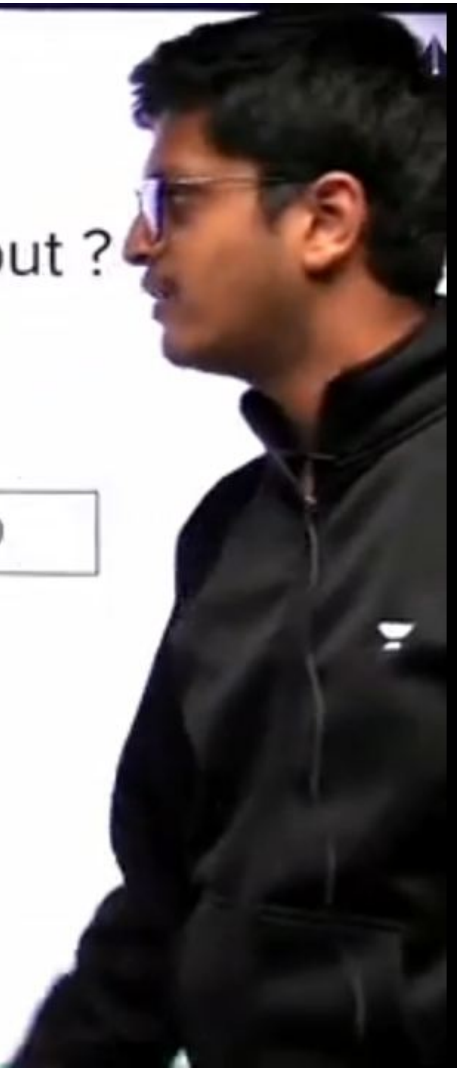
```
error: 'modify_a' is a private member of 'A'
error: 'a' is a private member of 'A'
```

Problem 1.b

```
#include<iostream>
using namespace std;
class A{
public:
    int a;
    void modify_a(int x){
        a = x;
    }
};
int main(){
    A a;
    a.modify_a(10);
    cout<<a.a<<endl;
    return 0;
}
```

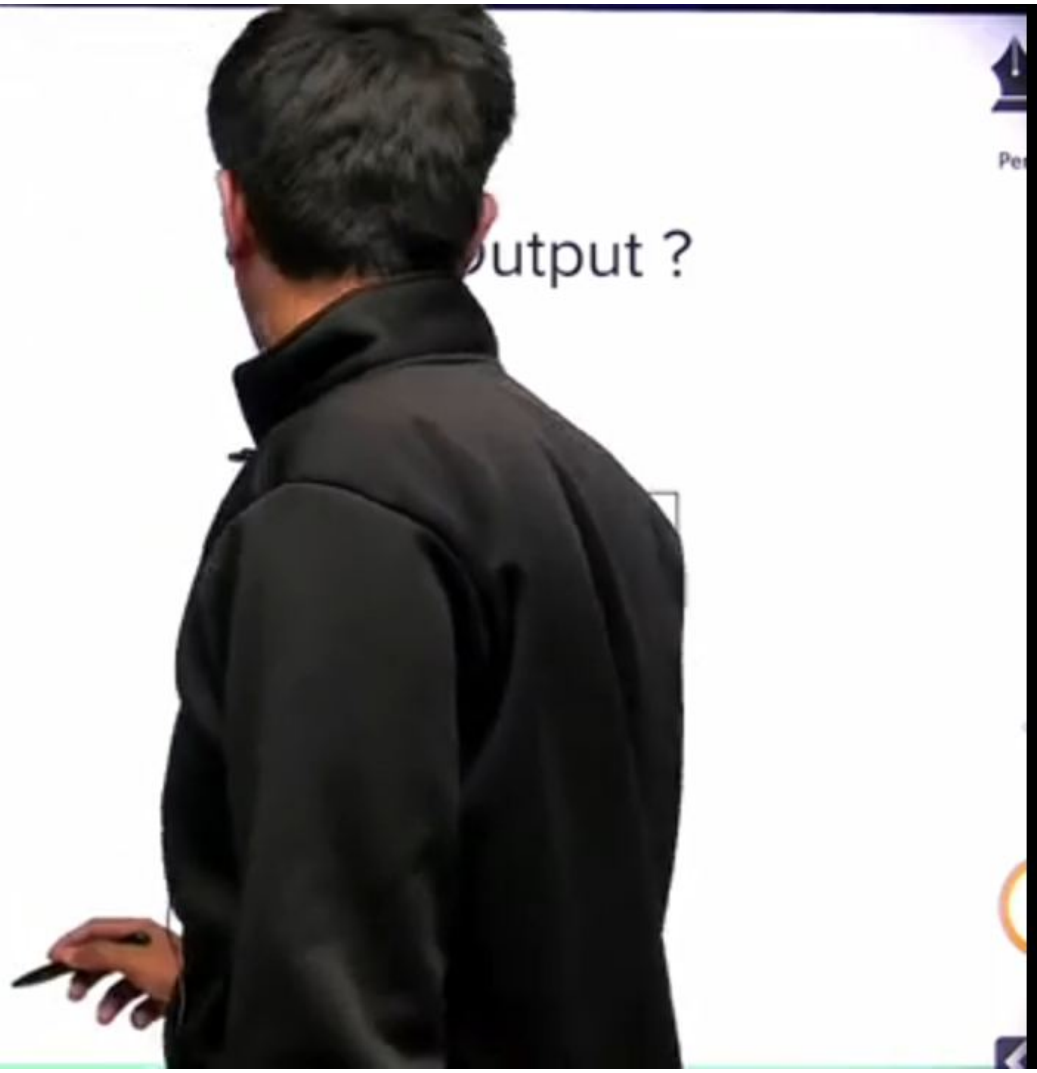
Output ?

10



Problem 1.c

```
#include<iostream>
using namespace std;
class A{
public:
    int a;
    A(int a){
        this->a = a;
    }
};
int main(){
    A a(10);
    cout<<a.a<<endl;
    return 0;
}
```



Problem 2.a

```
#include<iostream>
using namespace std;
class A{
public:
    int a, b;
    A(int a, int b){
        this->a = a; this ->b = b;
    }
    int getSum(){
        return (a+b);
    }
};
int main(){
    A obj1(3, 7);
    A *ptr;
    ptr = &obj1;
    *ptr.a = 5;
    cout<<obj1.getSum()<<endl;
    return 0;
}
```

Output ?

error: member reference type 'A *' is a pointer

Problem 2.c

```
#include<iostream>
using namespace std;
class A{
public:
    int a, b;
    A(int a, int b){
        this->a = a; this ->b = b;
    }
    int getSum(){
        return (a+b);
    }
};
int main(){
    A obj1(3, 7);
    A *ptr;
    ptr = &obj1;
    ptr->a = 5;
    cout<<obj1.getSum()<<endl;
    return 0;
}
```

Output ?

12

Problem 2.b

```
#include<iostream>
using namespace std;
class A{
public:
    int a, b;
    A(int a, int b){
        this->a = a; this ->b = b;
    }
    int getSum(){
        return (a+b);
    }
};
int main(){
    A obj1(3, 7);
    A *ptr;
    ptr = &obj1;
    (*ptr).a = 5;
    cout<<obj1.getSum()<<endl;
    return 0;
}
```

Output ?

12

Problem 3

```
#include<iostream>
using namespace std;
class A{
    int a;
public:
    A(){
        a = 5;
    }
    friend void modify();
};
void modify(){
    A obj1;
    cout<<"Private member of Class A, "<<obj1.a;
}
int main(){
    A obj1;
    //cout<<obj1.a<<endl;
    modify();
    return 0;
}
```

Output ?

Private member of Class A, 5

Problem 4

```cpp
#include <iostream>
using namespace std;
class Distance {
    private:
        int meter;
        friend int addFive(Distance);

    public:
        Distance() { meter = 0;}

};
int addFive(Distance d) {
    d.meter += 5;
    return d.meter;
}
int main() {
    Distance D;
    cout << "Distance: " << addFive(D);
    return 0;
}
```

Output ?

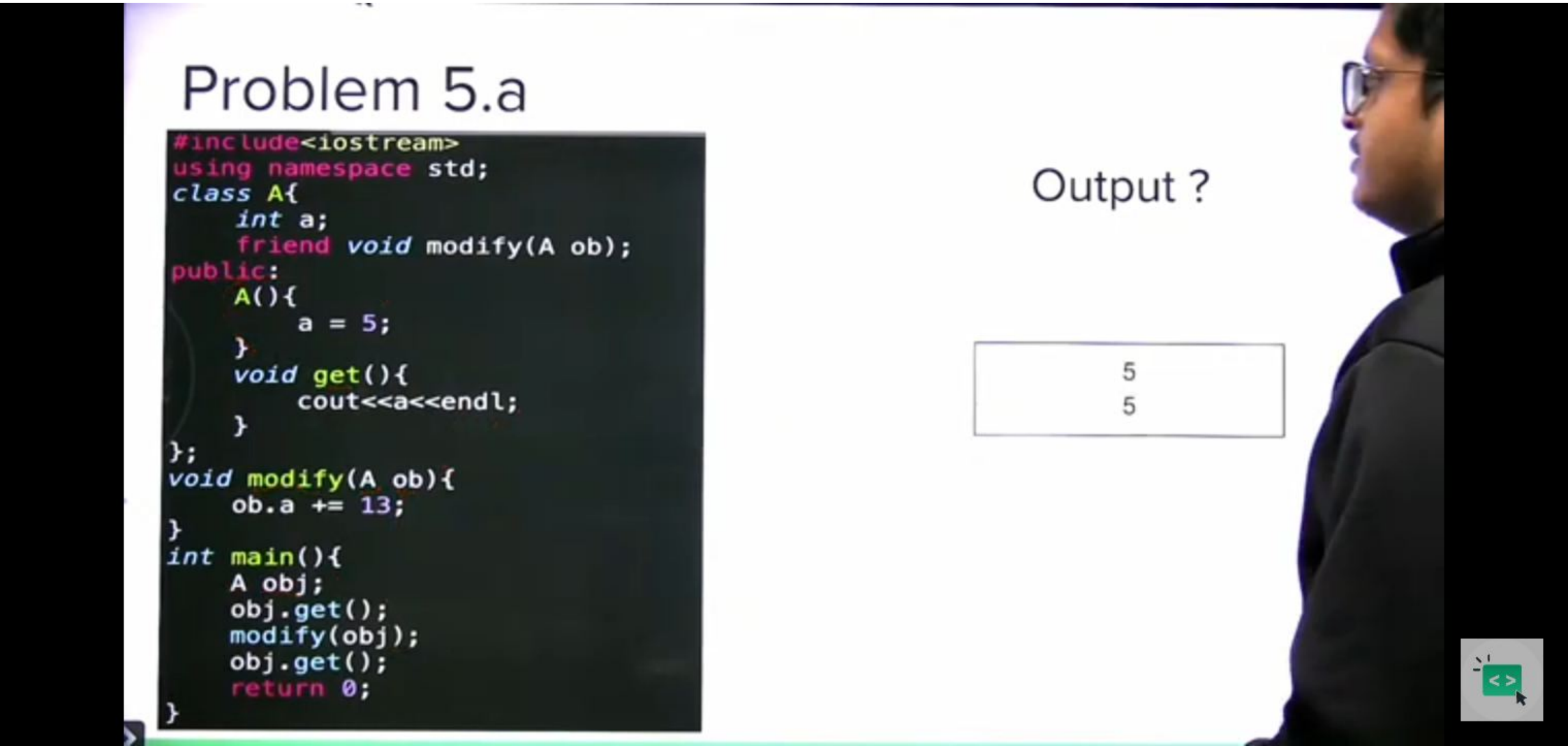
Distance: 5

Problem 5.a

```
#include<iostream>
using namespace std;
class A{
    int a;
    friend void modify(A ob);
public:
    A(){
        a = 5;
    }
    void get(){
        cout<<a<<endl;
    }
};
void modify(A ob){
    ob.a += 13;
}
int main(){
    A obj;
    obj.get();
    modify(obj);
    obj.get();
    return 0;
}
```

Output ?

```
5
5
```

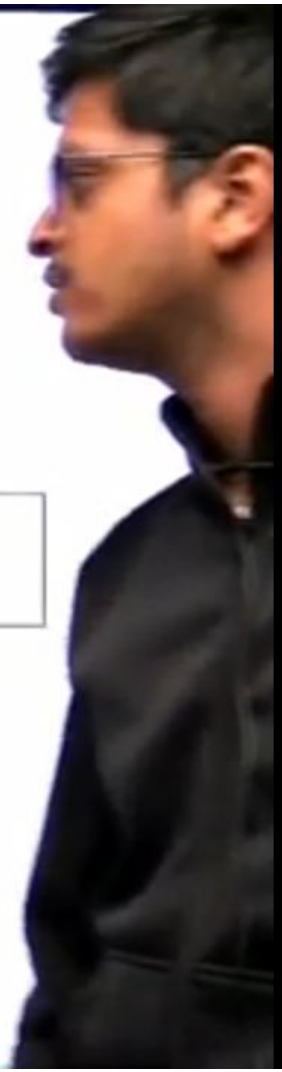


Problem 5.b

```
#include<iostream>
using namespace std;
class A{
    int a;
    friend void modify(A &ob);
public:
    A(){
        a = 5;
    }
    void get(){
        cout<<a<<endl;
    }
};
void modify(A &ob){
    ob.a += 13;
}
int main(){
    A obj;
    obj.get();
    modify(obj);
    obj.get();
    return 0;
}
```

Output ?

```
5
18
```



Problem 7

```
#include<iostream>
using namespace std;
class A{
public:
    static int a;
    void increment(){
        a++;
    }
    int get(){
        return a;
    }
};
int A::a = 10;
int main(){
    A obj1, obj2, obj3;
    obj1.increment();
    obj2.increment();
    obj3.increment();
    cout<<obj3.get()<<obj2.get()<<obj1.get()<<endl;
    return 0;
}
```

Output ?

131313

Problem 8

```
#include<iostream>
using namespace std;
class A{
public:
    static int a;
    static void increment(){
        a++;
    }
    static int get(){
        return a;
    }
};
int A::a = 10;
int main(){
    A obj1, obj2, obj3;
    obj1.increment();
    obj2.increment();
    A::increment();
    cout<<obj3.get()<<endl;
    return 0;
}
```

Output ?

13

Problem 9

```
class A{
public:
    static int a;
    static void increment(){
        a+=3;
    }
    int get(A obj){
        return (a+obj.a);
    }
};
int A::a = 10;
int main(){
    A obj1, obj2;
    obj1.increment(); obj2.increment();
    cout<<obj2.get(obj1)<<endl;
    return 0;
}
```

Output ?

32

Problem 10

```
#include<iostream>
using namespace std;
class A{
public:
    static int a;
    static void increment(){
        a+=3;
    }
    int get(A obj){
        return (a+obj.a);
    }
};
int A::a = 10;
int main(){
    A obj1;
    int *ptr;
    ptr = &obj1.a;
    *ptr *= 2;
    cout<<obj1.get(obj1)<<endl;
    return 0;
}
```

Output ?

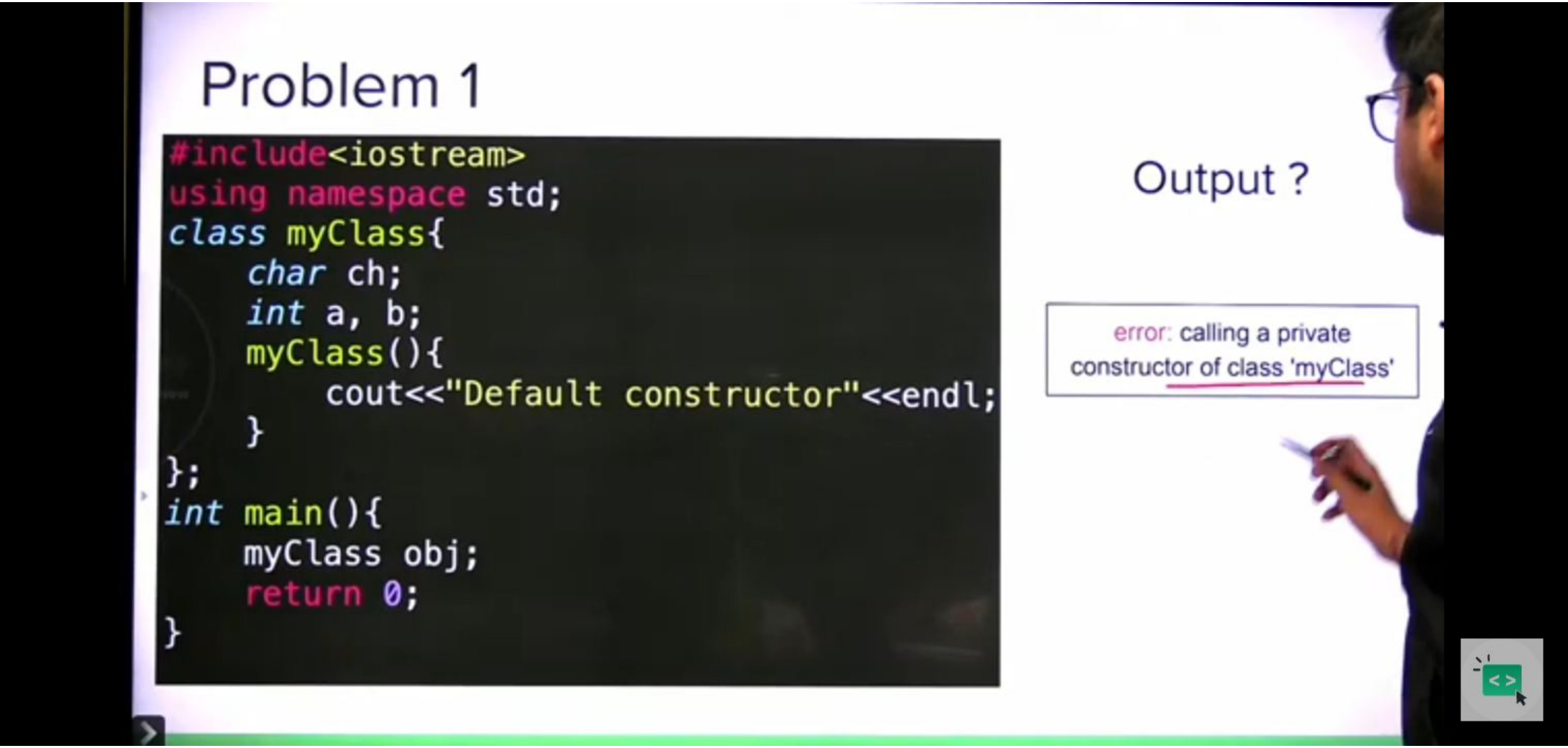
40

Problem 1

```
#include<iostream>
using namespace std;
class myClass{
    char ch;
    int a, b;
    myClass(){
        cout<<"Default constructor"<<endl;
    }
};
int main(){
    myClass obj;
    return 0;
}
```

Output ?

error: calling a private constructor of class 'myClass'



Problem 2

```
#include<iostream>
using namespace std;
class A{
private:
    A(){
        cout<<"Default Constructor"<<endl;
    }
public:
    int a;
    A(int a){
        this->a = a;
    }
};
int main(){
    A a(10);
    cout<<a.a<<endl;
    return 0;
}
```

Output ?

10

Problem 3

```
#include<iostream>
using namespace std;
class myClass{
    int a;
public:
    myClass(int x){
        a = x;
        cout<<"Constructor: "<<a<<endl;
    }
    int getvalue(){
        return a;
    }
};
int main(){
    myClass obj1(1), obj2;
    obj1.getvalue();
    obj2.getvalue();
    return 0;
}
```

Output ?

error: no matching constructor for initialization of 'myClass'

Problem 4

```
#include<iostream>
using namespace std;
class myClass{
public:
    char ch;
    int a, b;
    myClass(){
        a=0; b=0; ch='a';
    }
    myClass(char c){
        ch = c;
    }
    myClass(int x, int y){
        a = x;
        b = y;
    }
    void get(){
        cout<<a<<" "<<b<<" "<<ch<<endl;
    }
};
int main(){
    myClass ob1, ob2('Z'), ob3(4, 8);
    ob1.get();
    ob2.get();
    ob3.get();
    return 0;
}
```

Problem 5

```
#include<iostream>
using namespace std;
class myClass{
    int a;
public:
    myClass(int x){
        a = x;
        cout<<"Constructor: "<<a<<endl;
    }
    ~myClass(){
        cout<<"Destructor: "<<a<<endl;
    }
};
int main(){
    myClass obj1(1), obj2(2), obj3(3);
    return 0;
}
```

Output ?

```
Constructor: 1
Constructor: 2
Constructor: 3
Destructor: 3
Destructor: 2
Destructor: 1
```

Problem 6

```
#include<iostream>
using namespace std;
class myClass{
    int a;
public:
    myClass(int x){
        a = x;
        cout<<"Constructor: "<<a<<endl;
    }
    ~myClass(){
        cout<<"Destructor: "<<a<<endl;
    }
};
void myFun(int x){
    myClass obj1(x);
}
int main(){
    myClass obj1(1), obj2(2), obj3(3);
    myFun(4);
    myFun(5);
    return 0;
}
```

Output ?

Constructor: 1
Constructor: 2
Constructor: 3
Constructor: 4
Destructor: 4
Constructor: 5
Destructor: 5
Destructor: 3
Destructor: 2
Destructor: 1

Problem 6

```
#include<iostream>
using namespace std;
class myClass{
    int a;
public:
    myClass(int x){
        a = x;
        cout<<"Constructor: "<<a<<endl;
    }
    ~myClass(){
        cout<<"Destructor: "<<a<<endl;
    }
};
void myFun(int x){
    static myClass obj1(x);
}
int main(){
    myClass obj1(1), obj2(2), obj3(3);
    myFun(4);
    myFun(5);
    return 0;
}
```

Output ?

C 1
C 2
C 3
C 4
C 5
MFH
D3
D2
D1
D5
D4

Problem 7

```
#include <iostream>
#include <string>
using namespace std;
class A{
    int a;
public:
    A(){
        a = 10;
        cout<<"Default constructor: "<<a<<endl;
    }
    A(const A &q){
        a = q.a;
        cout<<"Copy Constructor: "<<a<<endl;
    }
    void set_a(int x){
        a = x;
    }
};
int main()
{
    A obj; obj.set_a(11);
    A a1 = obj;
    A a2(obj);
    return 0;
}
```

Ou

Default constru
Copy Construct
Copy Construct

Problem 8

```
#include<iostream>
using namespace std;
class A{
public:
    A(){
        cout<<"Constructor class A"<<endl;
    }
};
class myClass{
public:
    static A getvalue(){
        static A a;
        cout<<"Hello"<<endl;
        return a;
    }
};
int main(){
    myClass obj1;
    A obj2 = myClass::getvalue();
    A obj3 = obj1.getvalue();
    return 0;
}
```

Output ?

Constructor class A
Hello
Hello

Problem 9.a

```
class A{
public:
    A(){
        cout<<"A";
    }
    A(int x){
        cout<<x;
    }
};
class B{
    A obj1;
public:
    B(){
        cout<<"B";
    }
};
int main()
{
    B obj1;
    B obj2;
    return 0;
}
```

Output ?

ABAB

Problem 9.b

```
class A{
public:
    A(){
        cout<<"A";
    }
    A(int x){
        cout<<x;
    }
};
class B{
    A obj1;
public:
    static A ob;
    B(){
        cout<<"B";
    }
};
int main()
{
    B obj1;
    B obj2;
    return 0;
}
```

Output ?

ABAB

Problem 9.c

```
class A{
public:
    A(){
        cout<<"A";
    }
    A(int x){
        cout<<x;
    }
};
class B{
    A obj1;
public:
    static A ob;
    B(){
        cout<<"B";
    }
};
A B::ob;
int main()
{
    B obj1;
    B obj2;
    return 0;
}
```

Output ?

AABAB

Problem 10

```
class A{
    int a;
public:
    A(){
        a = 0;
    }
    A(int x){
        a = x;
    }
    void print(){
        cout<<a;
    }
};
int main()
{
    static A obj1;
    static A obj2(12);
    A a(11);
    A b = obj2;
    b.print();
    obj1 = a;
    obj1.print();
    return 0;
}
```

Output ?

obj1.a = 11
obj2.a = 12
a.a = 11
b.a = 12

12
11

Problem 1

```
#include<iostream>
using namespace std;
class A{
public:
    A(){
        cout<<"A";
    }
};
class B: public A{
public:
    B(){
        cout<<"B";
    }
};
int main(){
    B obj;
    return 0;
}
```

A B

Problem 1

```
#include<iostream>
using namespace std;
class A{
public:
    A(){
        cout<<"A";
    }
};
class B: public A{
public:
    B(){
        cout<<"B";
    }
};
int main(){
    B obj;
    return 0;
}
```

Output ?

AB

Problem 2

```
#include<iostream>
using namespace std;
class A{
public:
    A(int y){
        cout<<"A"<<y;
    }
};
class B: public A{
public:
    B(int x, int y): A(x){
        cout<<"B"<<y;
    }
};
int main(){
    B obj(4, 5);
    return 0;
}
```

Output ?

A4B5

Problem 3

```
#include<iostream>
using namespace std;
class A{
public:
    A(){
        cout<<"A";
    }
};
class C{
public:
    C(){
        cout<<"C";
    }
};
class B: public C, A{
public:
    B(){
        cout<<"B";
    }
};
int main(){
    B obj;
    return 0;
}
```

Output ?

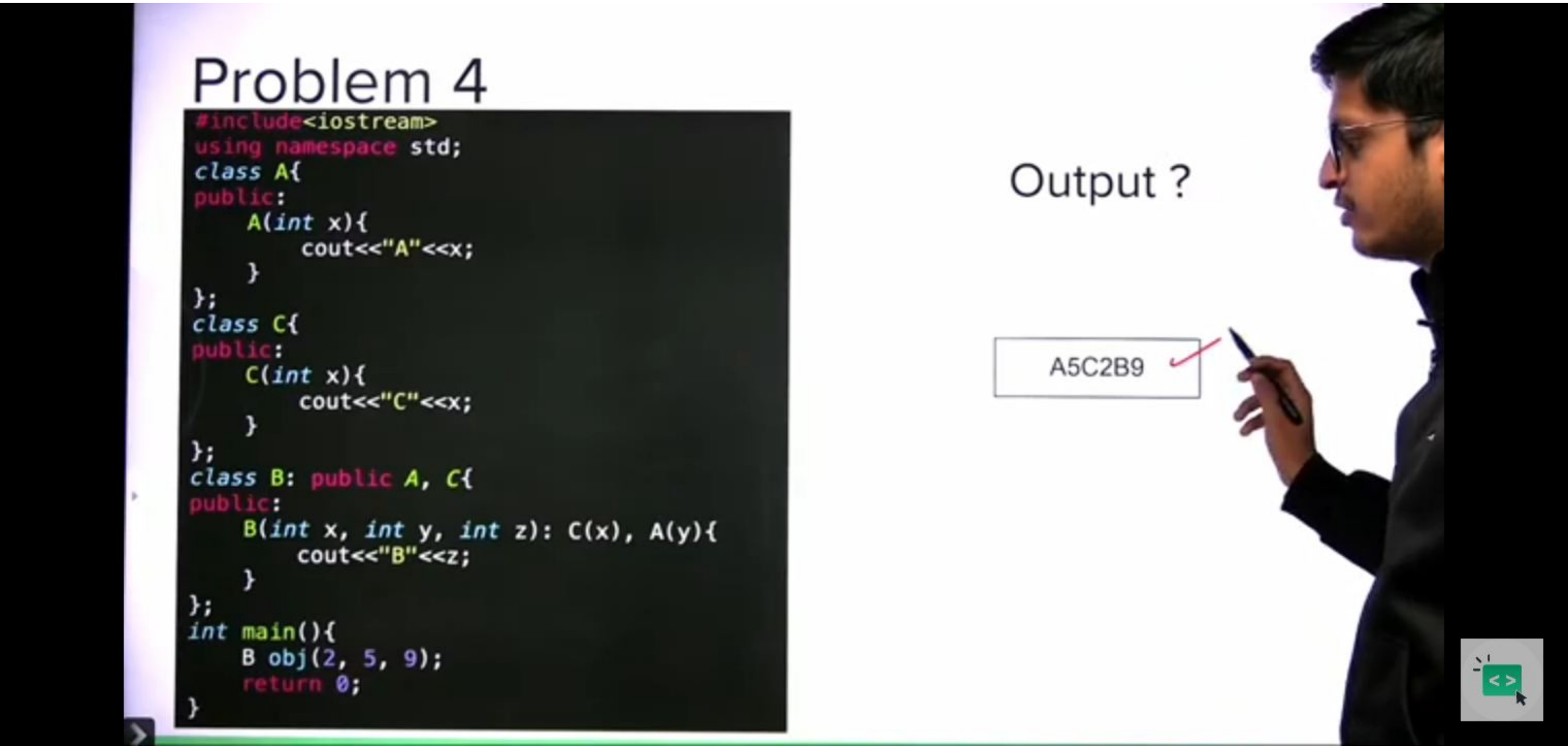
CAB

Problem 4

```
#include<iostream>
using namespace std;
class A{
public:
    A(int x){
        cout<<"A"<<x;
    }
};
class C{
public:
    C(int x){
        cout<<"C"<<x;
    }
};
class B: public A, C{
public:
    B(int x, int y, int z): C(x), A(y){
        cout<<"B"<<z;
    }
};
int main(){
    B obj(2, 5, 9);
    return 0;
}
```

Output ?

A5C2B9



Problem 5.a

```
#include<iostream>
using namespace std;
class A{
public:
    A(int x){
        cout<<"A"<<x;
    }
};
class B: public A{
public:
    B(int x){
        cout<<"B"<<x;
    }
};
int main(){
    B obj(12);
    return 0;
}
```

Output ?

error: constructor for 'B' must explicitly initialize the base class 'A' which does not have a default constructor

Problem 5.b

```
#include<iostream>
using namespace std;
class A{
public:
    A(int x){
        cout<<"A"<<x;
    }
};
class B: public A{
public:
    B(int x): A(x){
        cout<<"B"<<x;
    }
};
int main(){
    B obj(12);
    return 0;
}
```

A12 B12

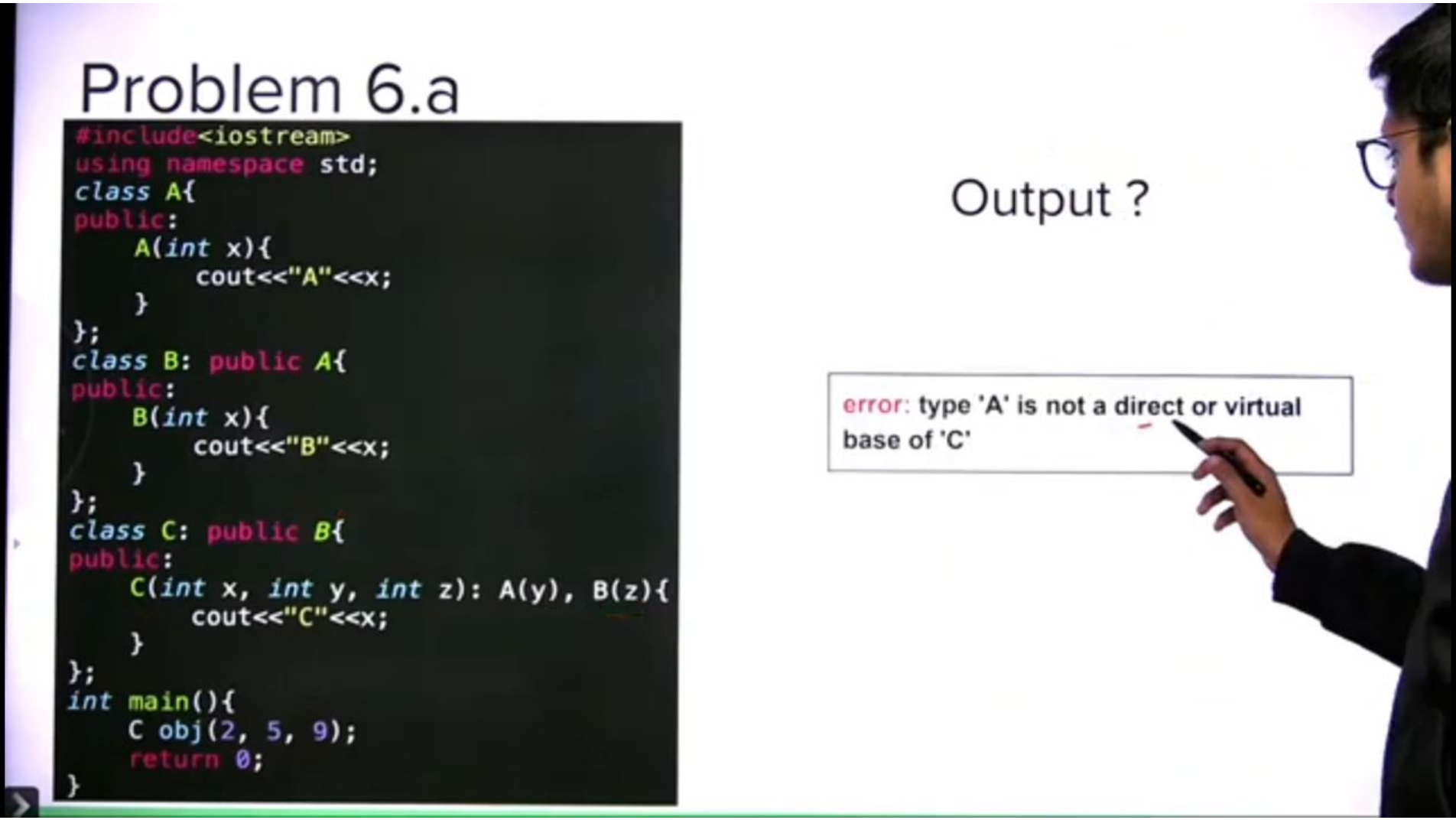
Output ?

Problem 6.a

```
#include<iostream>
using namespace std;
class A{
public:
    A(int x){
        cout<<"A"<<x;
    }
};
class B: public A{
public:
    B(int x){
        cout<<"B"<<x;
    }
};
class C: public B{
public:
    C(int x, int y, int z): A(y), B(z){
        cout<<"C"<<x;
    }
};
int main(){
    C obj(2, 5, 9);
    return 0;
}
```

Output ?

error: type 'A' is not a direct or virtual base of 'C'



Problem 6.b

```
#include<iostream>
using namespace std;
class A{
public:
    A(int x){
        cout<<"A"<<x;
    }
};
class B: public A{
public:
    B(int x, int y): A(y){
        cout<<"B"<<x;
    }
};
class C: public B{
public:
    C(int x, int y, int z): B(y, z){
        cout<<"C"<<x;
    }
};
int main(){
    C obj(2, 5, 9);
    return 0;
}
```

Output ?

A 9
B 5
C 2

Problem 6.c

```
class A{
public:
    A(int x){
        cout<<"A"<<x;
    }
    void get(){cout<<" Hello world ";}
};
class B: public A{
public:
    B(int x, int y): A(y){
        cout<<"B"<<x;
    }
};
class C: public B{
public:
    C(int x, int y, int z): B(y, z){
        get();
        cout<<"C"<<x;
    }
};
int main(){
    C obj(2, 5, 9);
    return 0;
}
```

Output ?

A 9 B 5
Hello world
C 2

Problem 7.a

```
class A{
public:
    A(int x){
        cout<<"A"<<x;
    }
    void get(){cout<<" Hello world ";}
};
class B: protected A{
public:
    B(int x, int y): A(y){
        cout<<"B"<<x;
    }
};
class C: public B{
public:
    C(int x, int y, int z): B(y, z){
        get();
        cout<<"C"<<x;
    }
};
```

Output ?

C obj(2, 5, 9);

A9B5 Hello world C2

Problem 7.b

```
class A{
public:
    A(int x){
        cout<<"A"<<x;
    }
    void get(){cout<<" Hello world ";}
};
class B: protected A{
public:
    B(int x, int y): A(y){
        cout<<"B"<<x;
    }
};
class C: public B{
public:
    C(int x, int y, int z): B(y, z){
        get();
        cout<<"C"<<x;
    }
};
```

Output ?

```
A a(20);
a.get();
```

A20 Hello world

Problem 8

```
class A{
public:
    ~A(){
        cout<<"A";
    }
};
class B{
public:
    ~B(){
        cout<<"B";
    }
};
class C: public A, B{
public:
    ~C(){
        cout<<"C";
    }
};
```

Output ?

C c;

CBA

Problem 9

```
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);
  return 0;
}
```

Output ?

Assume :
sizeof(int) = 4B
sizeof(class) = sum of sizeof all attributes in it

Problem 9

```
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);
  return 0;
}
```

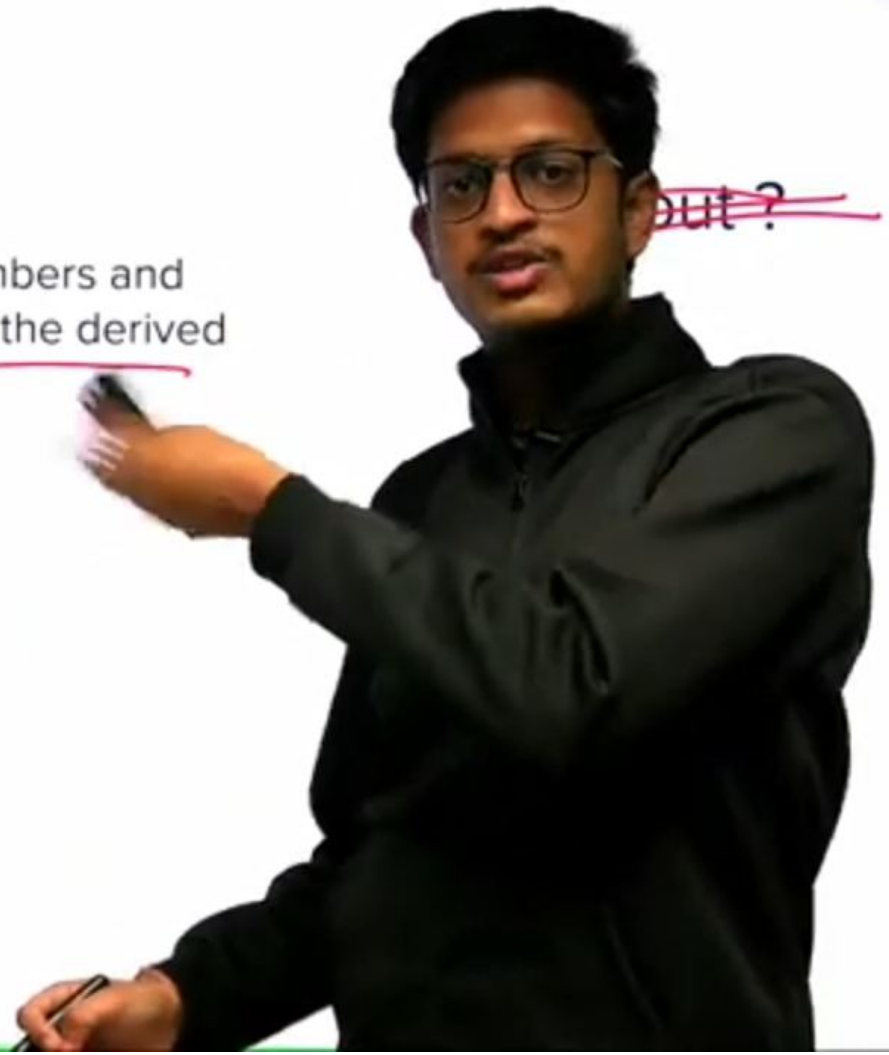
Output ?

80

Problem 10.a

Which specifier makes all the data members and functions of base class inaccessible by the derived class?

a. private
b. protected
c. public
d. both private and protected



Problem 10.b

```
class A{
public:
    int a;
    void change(int i){
        a = i;
    }
    void value_of_a(){
        cout<<a;
    }
};
class B: public A{
    int a = 15;
public:
    void print(){
        cout<<a;
    }
};
```

Output ?

```
B b;
b.change(10);
b.print();
b.value_of_a();
```

1510

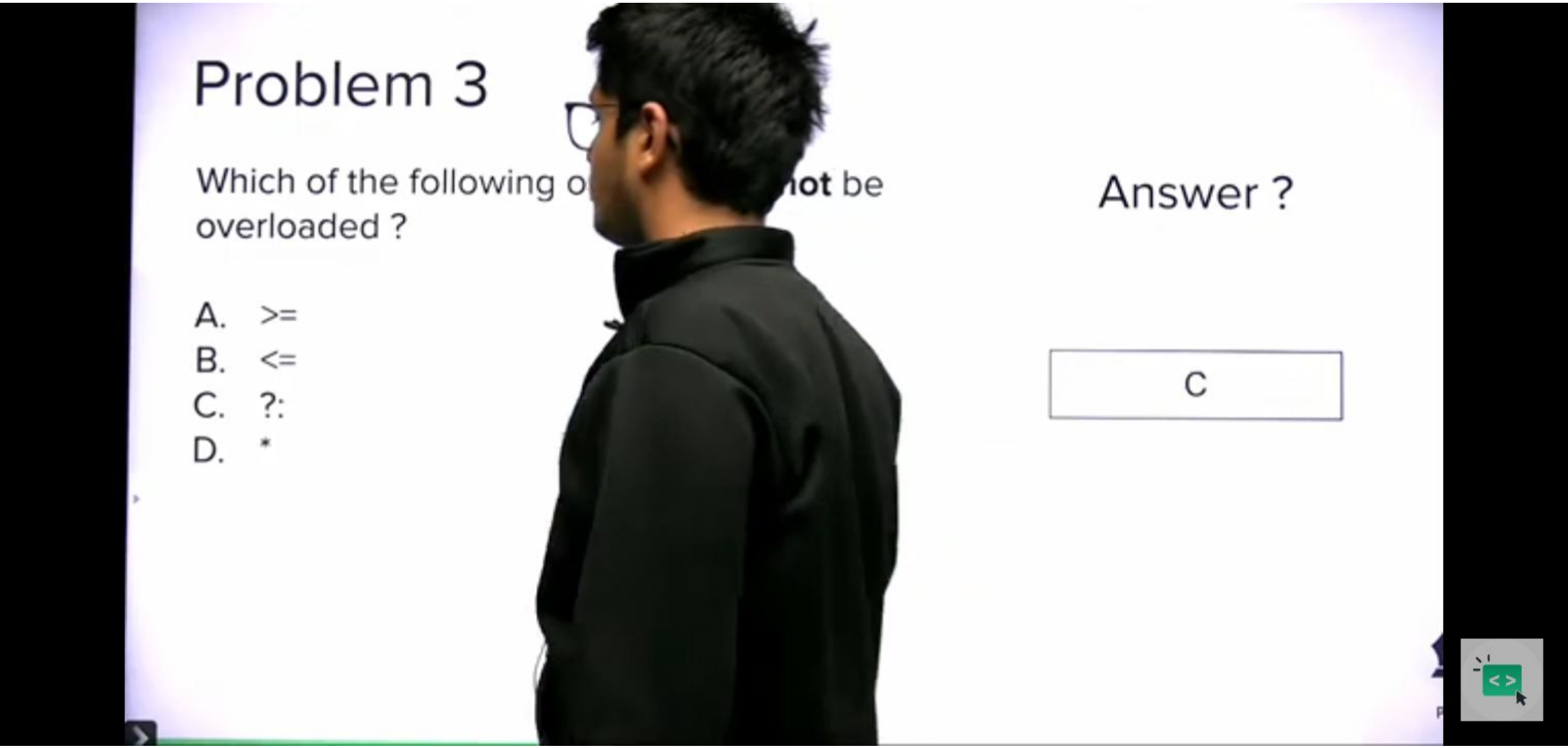
Problem 2

Which of the following operator **can be** overloaded ?

A. :: (scope resolution)
B. . (member selection)
C. += (shorthand assignment)
D. .* (member selection through pointer to function)
E. sizeof(datatype)

Answer ?

C

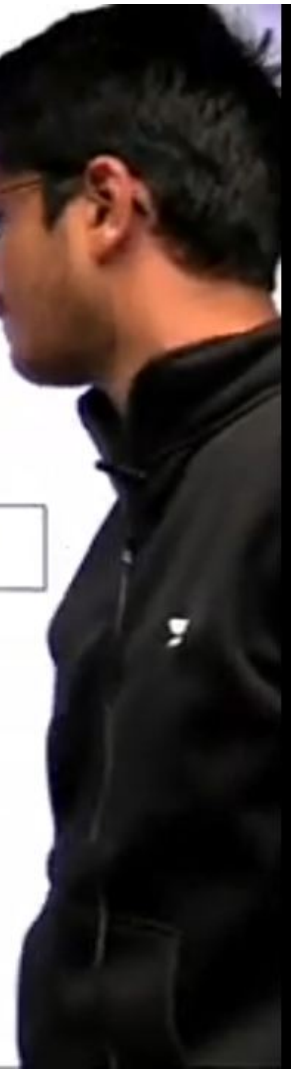

Problem 3
Which of the following o ot be overloaded ?
A. >=
B. <=
C. ?:
D. *
Answer ?
C

Problem 4

```
class A{
public:
    void function(){
        cout<<"A";
    }
};
class B{
public:
    void function(){
        A obj;
        cout<<"B";
        obj.function();
    }
};
int main(){
    B b;
    b.function();
    return 0;
}
```

Output ?

BA



Problem 5

```
class A{
public:
    int function(int x){
        return 2*x;
    }
    int function(int x, int y){
        return x+10*y;
    }
    char function(char x){
        return x+1;
    }
    double function(double x){
        return x;
    }
};
int main(){
    A a;
    cout<<a.function(123);
    cout<<a.function('a');
    cout<<a.function(12.4);
    cout<<a.function(4, 5);
    return 0;
}
```

Output

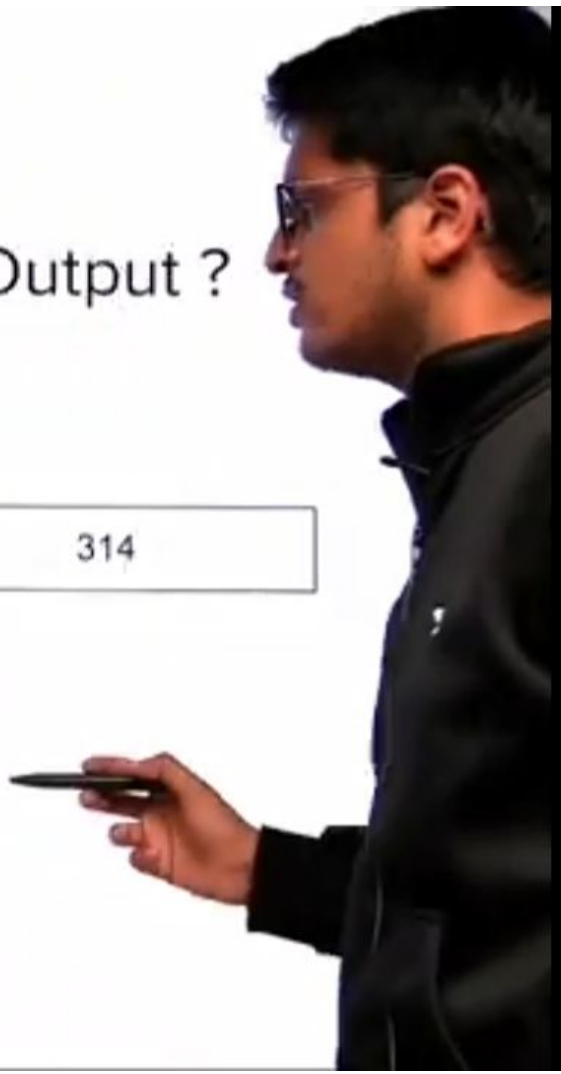
246b12.454

Problem 6

```
class A{
    int a, b;
public:
    A(int x, int y)  {a = x;  b = y;}
    A operator+(A const &obj){
        A temp(0,0);
        temp.a = a + obj.a;
        temp.b = b + obj.b;
        return temp;
    }
    void print(){
        cout<<a<<b<<endl;
    }
};

int main()
{
    A obj1(2, 6), obj2(1, 8);
    A obj3 = obj1+obj2;
    obj3.print();
}
```

Output ?

314



Problem 7

```
class A{
    int a, b;
public:
    A(int x, int y)  {a = x;  b = y;}
    void operator+(A const &obj){
         a += obj.a;
         b += obj.b;
    }
    void print(){
        cout<<a<<b<<endl;
    }
};

int main()
{
    A obj1(2, 6), obj2(1, 8);
    obj1+obj2;
    obj1.print();
    obj2.print();
}
```

Output ?

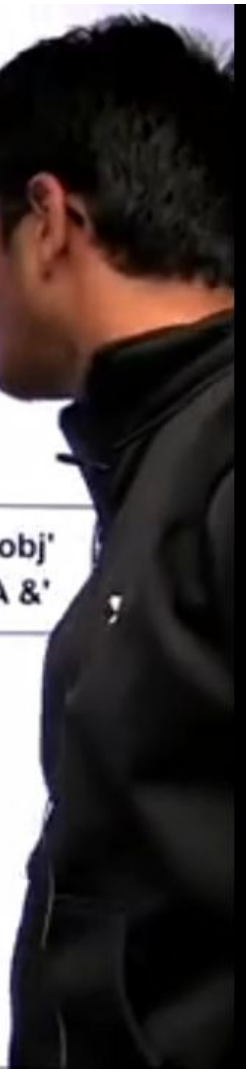
```
314
18
```

Problem 8

```
class A{
    int a, b;
public:
    A(int x, int y)  {a = x;  b = y;}
    void operator+(A const &obj){
         obj.a += a;
         obj.b += b;
    }
    void print(){
        cout<<a<<b<<endl;
    }
};

int main()
{
    A obj1(2, 6), obj2(1, 8);
    obj1+obj2;
    obj1.print();
    obj2.print();
}
```

Output ?

error: cannot assign to variable 'obj'
with const-qualified type 'const A &'



Problem 9

```
class A{
public:
    int add(int x, int y){
        return x+y;
    }
    int add(int x, int y, int z){
        return x+y+z;
    }
    int add(int *arr, int n){
        int sum = 0;
        for(int i=0; i<n; i++){
            sum += arr[i];
        }
        return sum;
    }
};

int main()
{
    int a=3, b=9, c=1;
    int arr[] = {1, 2, 3, 4};
    A obj;
    cout<<obj.add(a, b)<<endl;
    cout<<obj.add(a, b, c)<<endl;
    cout<<obj.add(arr, 4);
}
```

Output ?

12 ✓
13 ✓
10 ✓

Problem 11

```
class A{
public:
    int add(int x, int y, int z=0){
        return x+y+z;
    }
    int add(int x, int y){
        return x+y;
    }
};
int main()
{
    int a=2, b=16, c=1;
    A obj;
    cout<<obj.add(a, b, c)<<endl;
    cout<<obj.add(a, b)<<endl;
    return 0;
}
```

Output

Problem 10

```
class A{
public:
    int add(int x=0, int y=0){
        return x+y+20;
    }
    int add(int x, int y){
        return x+y;
    }
};
int main()
{
    int a=2, b=16;
    A obj;
    cout<<obj.add()<<endl;
    cout<<obj.add(a, b)<<endl;
    return 0;
}
```

Output

error: class member cann

Problem 11

```
class A{
public:
    int add(int x, int y, int z=0){
        return x+y+z;
    }
    int add(int x, int y){
        return x+y;
    }
};
int main()
{
    int a=2, b=16, c=1;
    A obj;
    cout<<obj.add(a, b, c)<<endl;
    cout<<obj.add(a, b)<<endl;
    return 0;
}
```

Output ?

```
error: call to member function 'add' is ambiguous
```

Problem 12

```
#include<iostream>
using namespace std;
class A{
public:
    int fun(int x){
        return x*x;
    }
    int fun(int &x){
        return x*10;
    }
};
int main(){
    int a=2;
    A obj;
    cout<<obj.fun(a);
    return 0;
}
```

Output ?

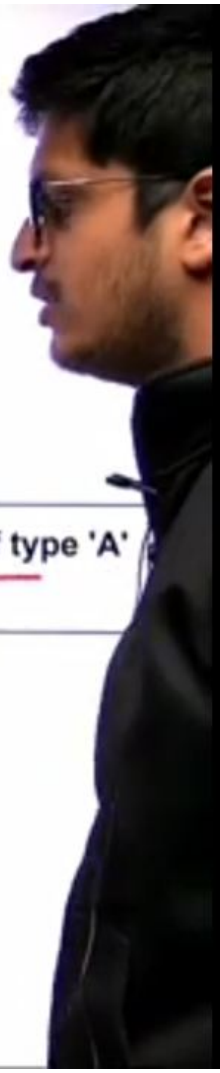
error: call to member function 'fun' is ambiguous

Problem 13

```
class A{
    int a;
public:
    A(){
        a = 0;
    }
    void operator++(){
        a+=3;
    }
    void print(){
        cout<<a<<endl;
    }
};
int main(){
    A obj;
    for(int i=0; i<5; i++) obj++;
    obj.print();
    return 0;
}
```

Output ?

error: cannot increment value of type 'A'

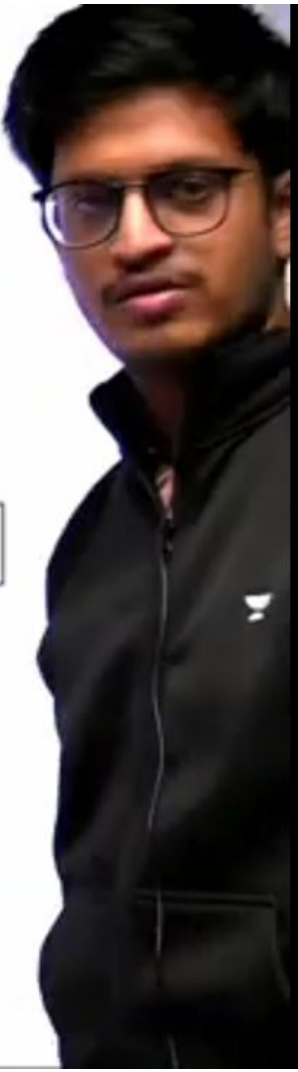


Problem 14

```
class A{
    int a;
public:
    A(){
        a = 0;
    }
    void operator++(){
        a+=3;
    }
    void print(){
        cout<<a<<endl;
    }
};
int main(){
    A obj;
    for(int i=0; i<5; i++) ++obj;
    obj.print();
    return 0;
}
```

Output ?

15

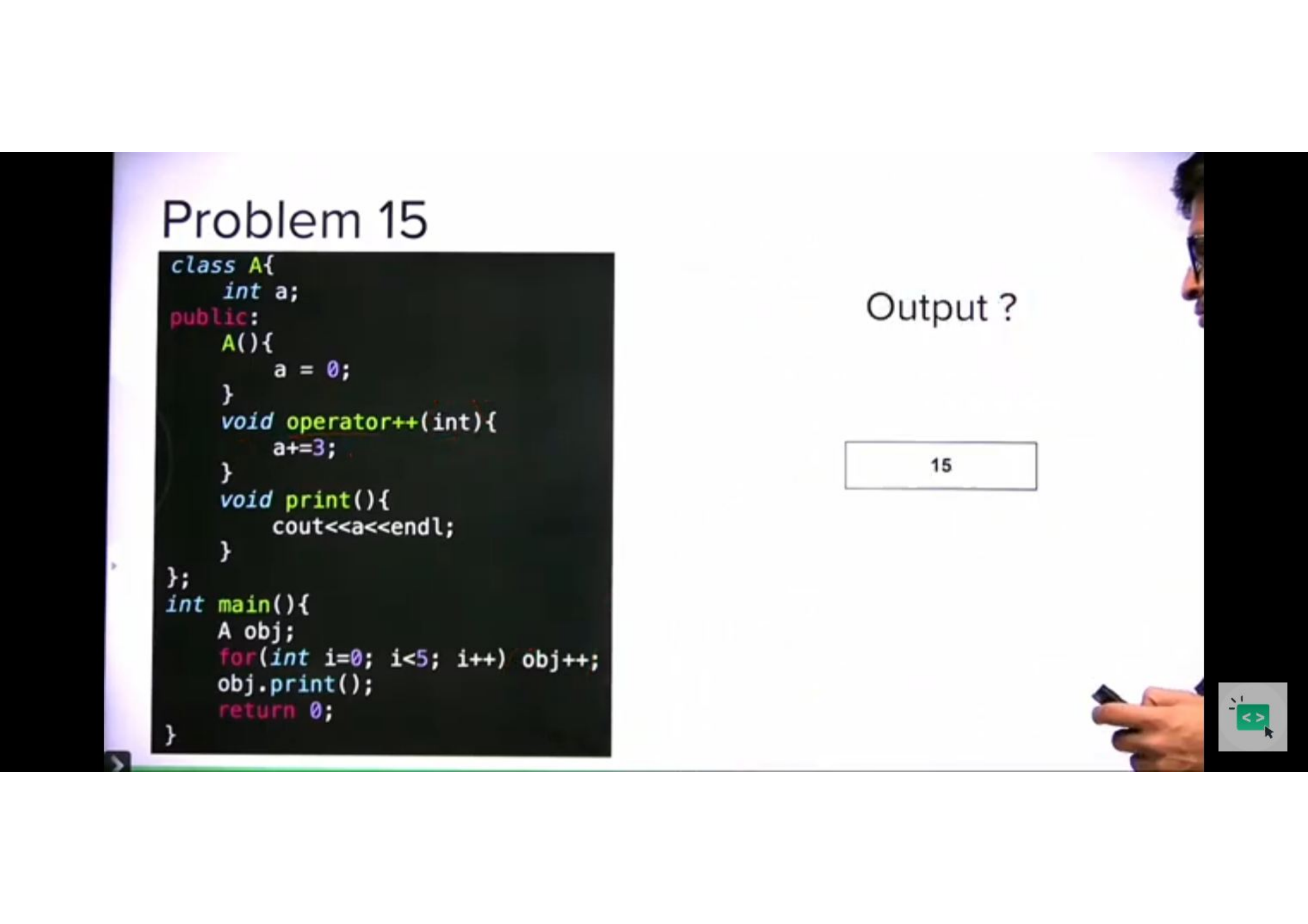


Problem 15

```
class A{
    int a;
public:
    A(){
        a = 0;
    }
    void operator++(int){
        a+=3;
    }
    void print(){
        cout<<a<<endl;
    }
};
int main(){
    A obj;
    for(int i=0; i<5; i++) obj++;
    obj.print();
    return 0;
}
```

Output ?

15

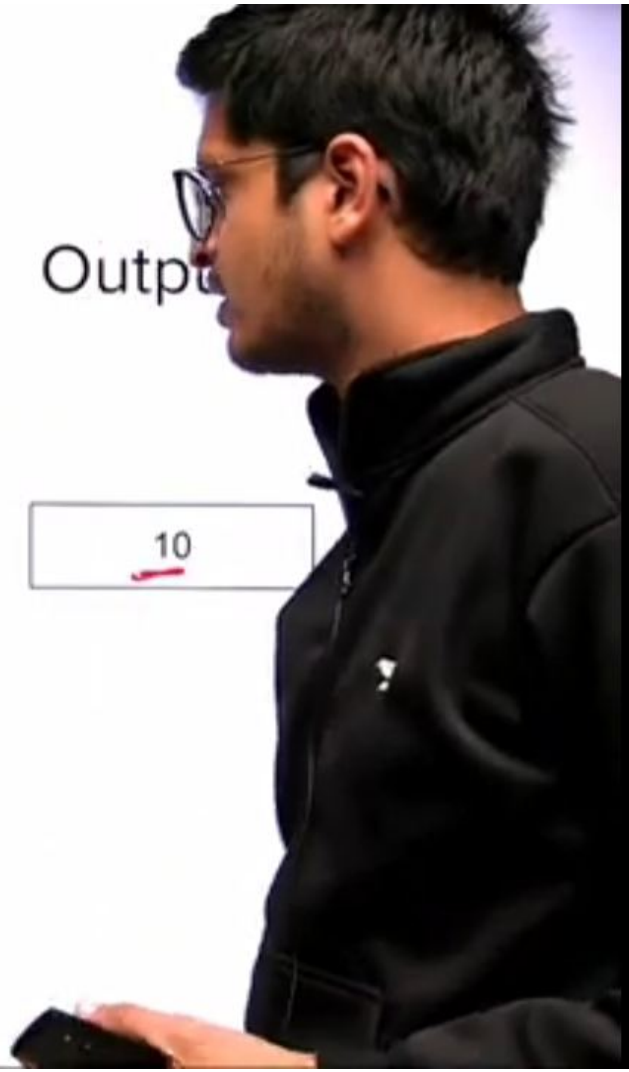


Problem 16

```
class A{
    int a, b;
public:
    A(){
        a = 0, b = 0;
    }
    void set(int x, int y = 4){
        a = x; b = y;
    }
    bool operator==(const A& obj){
        if(a==(obj.a+2) && b==obj.b) return true;
        else return false;
    }
};
int main(){
    A obj1, obj2;
    obj1.set(4, 6); obj2.set(2, 6);
    cout<<(obj1==obj2);
    obj2.set(6, 6);
    cout<<(obj1==obj2);
    return 0;
}
```

Outpt

10

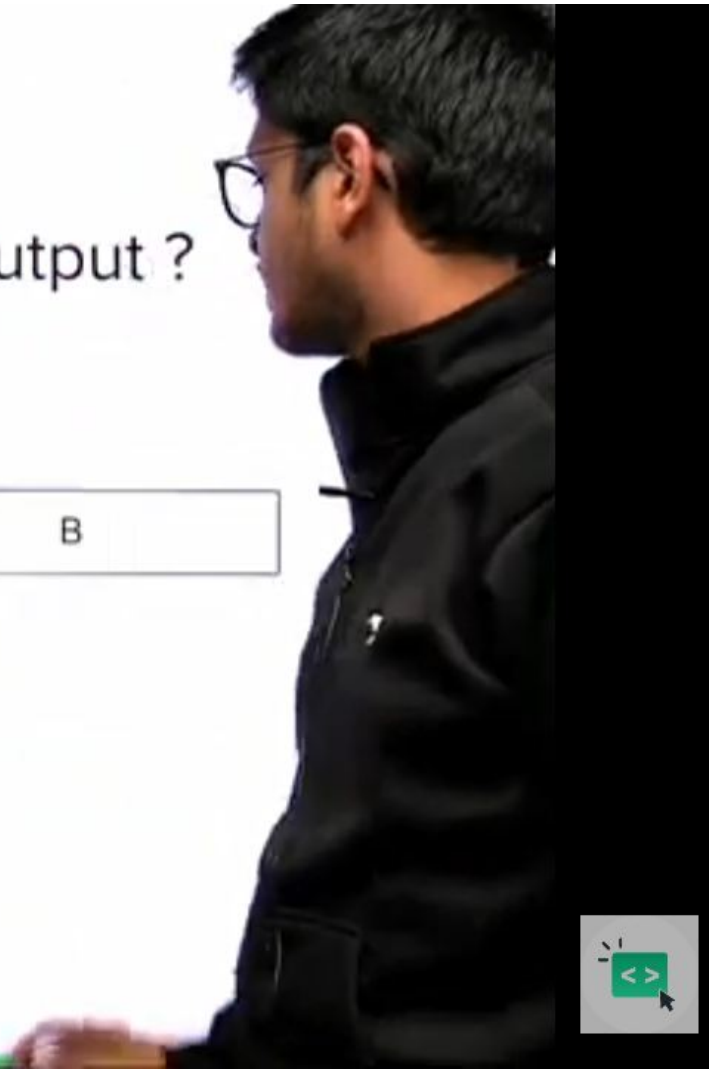


Problem 1

```
class A{
public:
    void function(){
        cout<<"A";
    }
};
class B: public A{
public:
    void function(){
        cout<<"B";
    }
};
int main(){
    B b;
    b.function();
    return 0;
}
```

Output ?

B

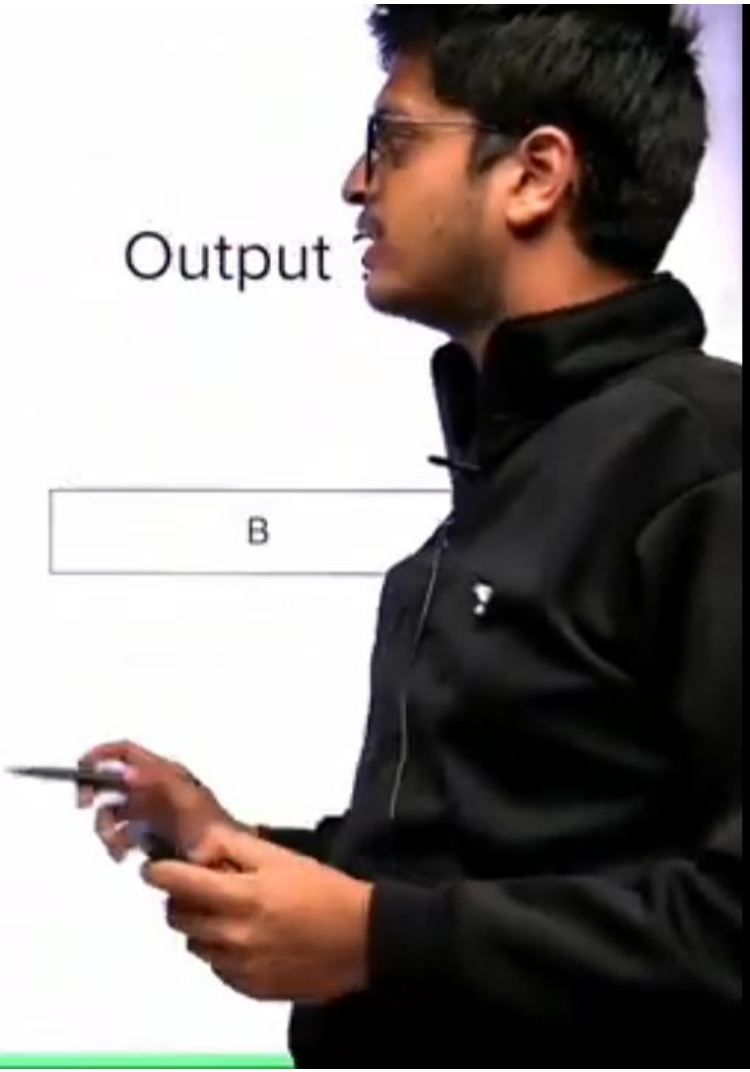


Problem 2

```
class A{
public:
    void function(){
        cout<<"A";
    }
};
class C{
public:
    void function(){
        cout<<"C";
    }
};
class B: public A, C{
public:
    void function(){
        cout<<"B";
    }
};
int main(){
    B b;
    b.function();
    return 0;
}
```

Output

B

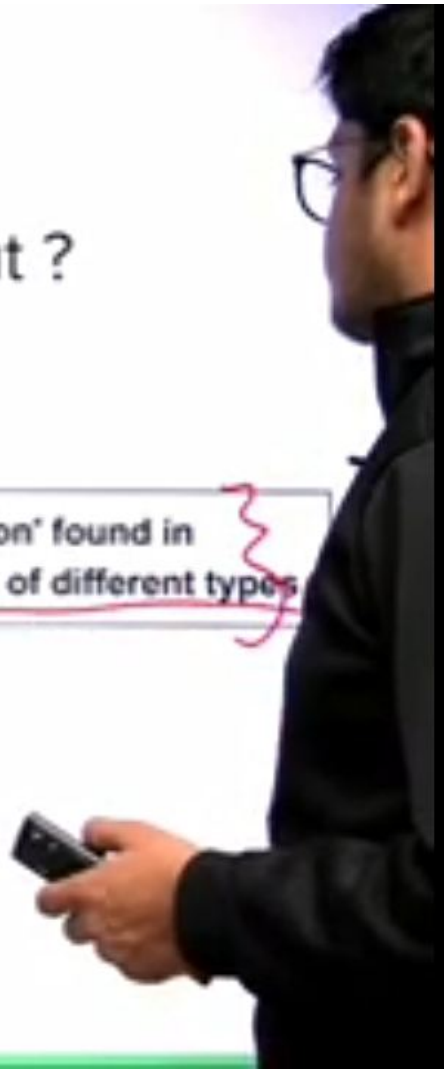


Problem 3

```
class A{
public:
    void function(){
        cout<<"A";
    }
};
class C{
public:
    void function(){
        cout<<"C";
    }
};
class B: public A, C{
public:
    void fun2(){
        function();
        cout<<"B";
    }
};
int main(){
    B b;
    b.fun2();
    return 0;
}
```

Output ?

error: member 'function' found in multiple base classes of different types

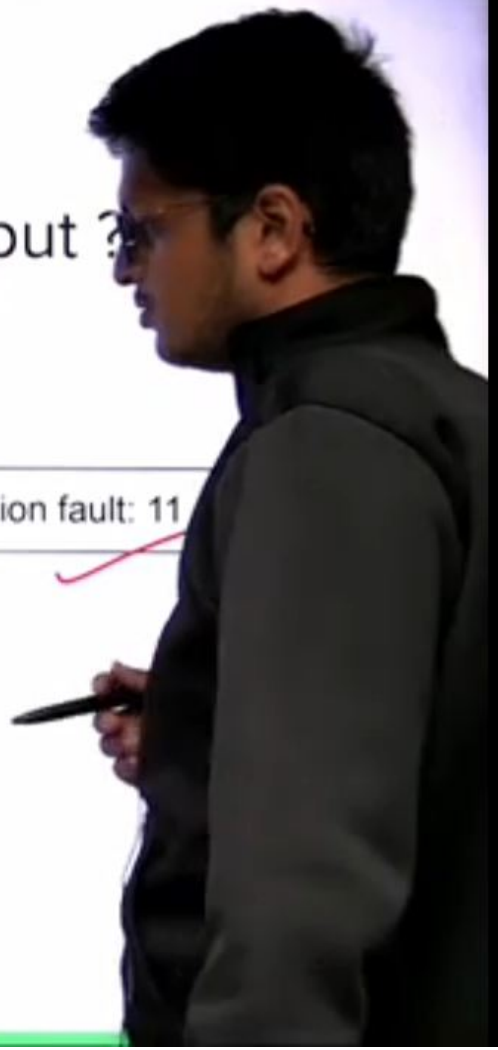


Problem 4

```
class A{
public:
    void function(){
        cout<<"A";
    }
};
class C{
public:
    void function(){
        cout<<"C";
    }
};
class B: public A, C{
public:
    void function(){
        function();
        cout<<"B";
    }
};
int main(){
    B b;
    b.function();
    return 0;
}
```

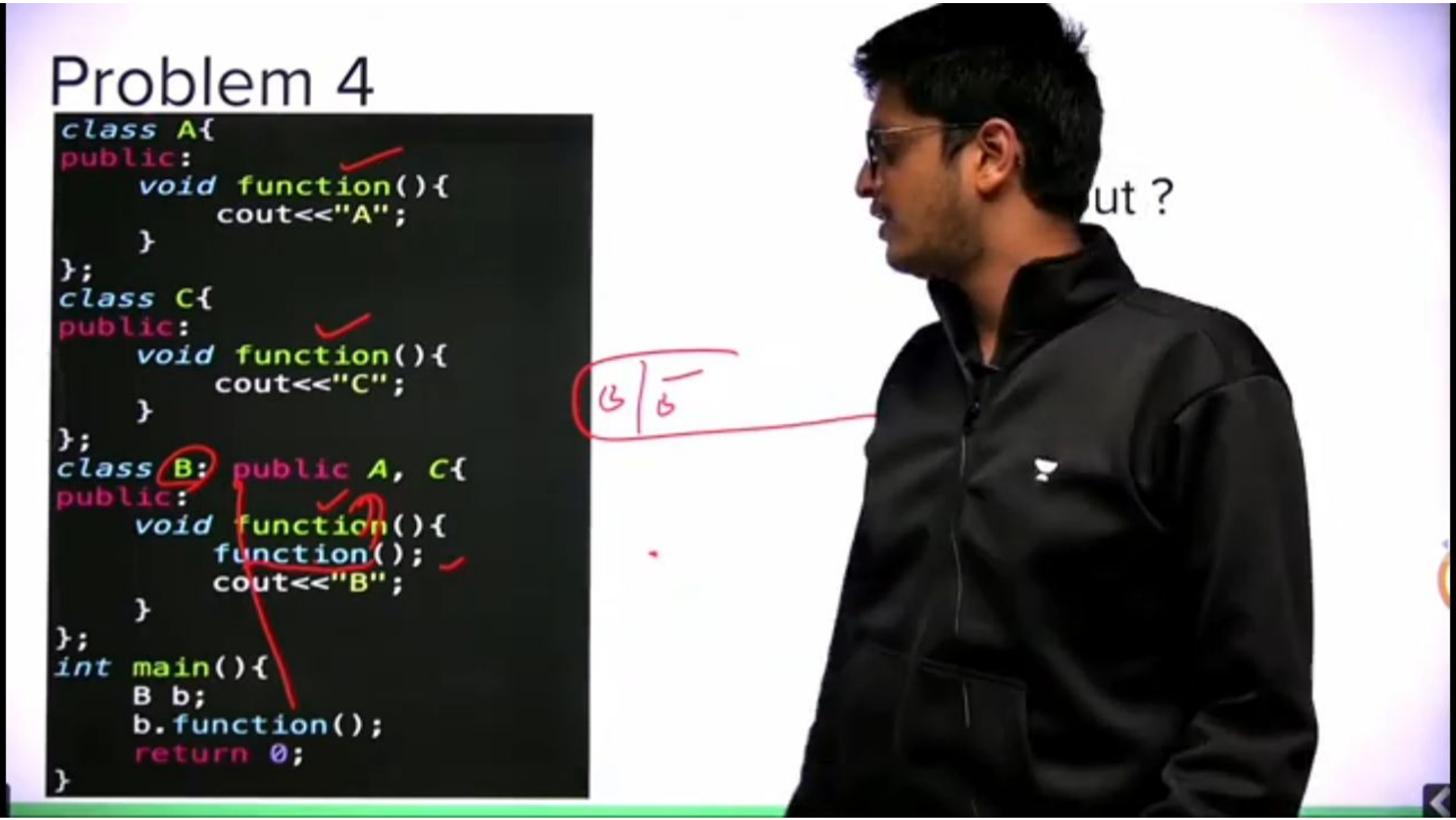
Output ?

Segmentation fault: 11



Problem 4

```
class A{
public:
    void function(){
        cout<<"A";
    }
};
class C{
public:
    void function(){
        cout<<"C";
    }
};
class B: public A, C{
public:
    void function(){
        function();
        cout<<"B";
    }
};
int main(){
    B b;
    b.function();
    return 0;
}
```



Problem 1

```
class B{
    int a;
public:
    void fun(){
        cout<<"Base class"<<endl;
    }
};
class D: public B{
    int a;
public:
    void fun(){
        cout<<"Derived class"<<endl;
    }
};
```

Output

```
B b = B();
b.fun();
```

```
Base class
```

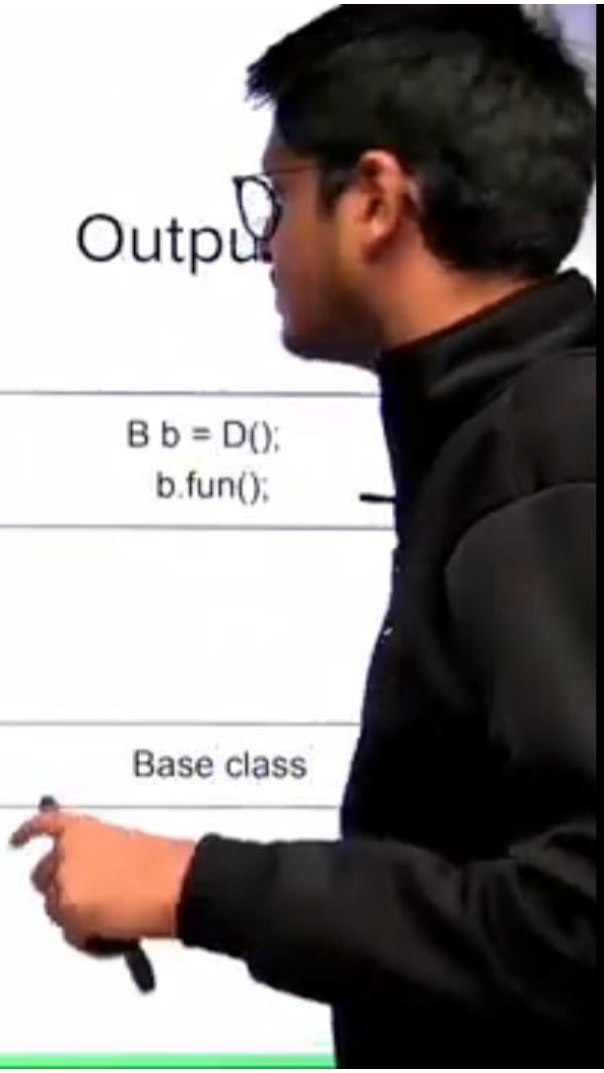
Problem 2

```
class B{
    int a;
public:
    void fun(){
        cout<<"Base class"<<endl;
    }
};
class D: public B{
    int a;
public:
    void fun(){
        cout<<"Derived class"<<endl;
    }
};
```

Outpu

```
B b = D();
b.fun();
```

Base class



Problem 3

```
class B{
    int a;
public:
     virtual void fun(){
         cout<<"Base class"<<endl;
    }
};
class D: public B{
    int a;
public:
    void fun(){
         cout<<"Derived class"<<end
    }
};
int main(){
    B obj1;
    D obj2;
    B *b = &ob
    b->fun();
    return 0;
}
```

Output ?

Base class

Problem 4

```
class B{
    int a;
public:
     virtual void fun(){
         cout<<"Base class"<<endl;
    }
};
class D: public B{
    int a;
public:
    void fun(){
         cout<<"Derived class"<<endl;
    }
};
int main(){
    B obj1;
    D obj2;
    B *b = &obj2;
    b->fun();
    return 0;
}
```

Out

Derived class

Problem 6

```
class B{
    int a;
public:
     virtual void fun(){
         cout<<"Base class"<<endl;
    }
};
class D: public B{
    int a;
public:
    void fun(){
         cout<<"Derived class"<<endl;
    }
};
int main(){
    B obj1;
    D obj2;
    D *b = &obj1;
    b->fun();
    return 0;
}
```

Output ?

error: cannot initialize a variable of type 'D *' with an rvalue of type 'B *'

Problem 7

```
class B{
    int a;
public:
     static virtual void fun(){
         cout<<"Base class"<<endl;
    }
};
class D: public B{
    int a;
public:
    void fun(){
         cout<<"Derived class"<<endl;
    }
};
int main(){
    B obj1;
    D obj2;
    B *b = &obj1;
    b->fun();
    return 0;
}
```

Outpu

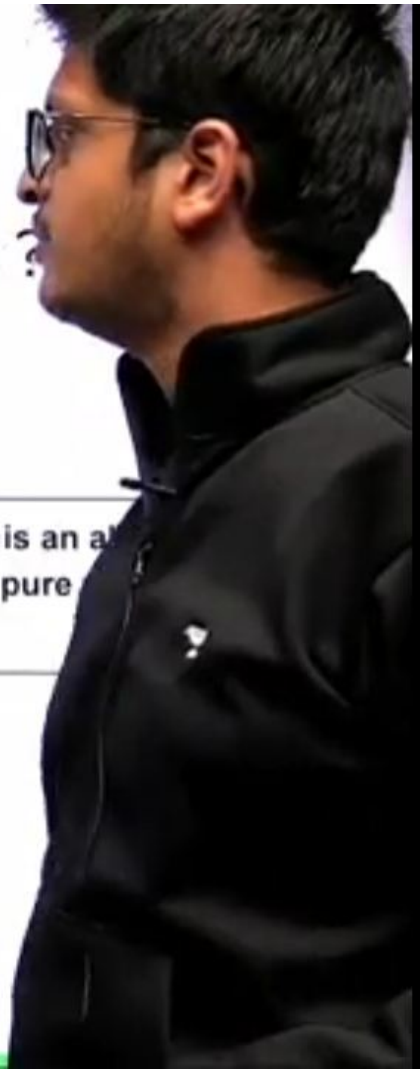
error: 'virtual' can only ap
non-static member functi

Problem 8.b

```
class B{
    int a;
public:
    virtual void fun()=0;
    void fun2(){
        cout<<"Hello world"<<endl;
    }
};
class D: public B{
public:
    D(){
        cout<<"B"<<endl;
    }
};
int main(){
    D obj1;
    obj1.fun();
    return 0;
}
```

Output ?

error: variable type 'D' is an a
class, unimplemented pure
method 'fun' in 'D'



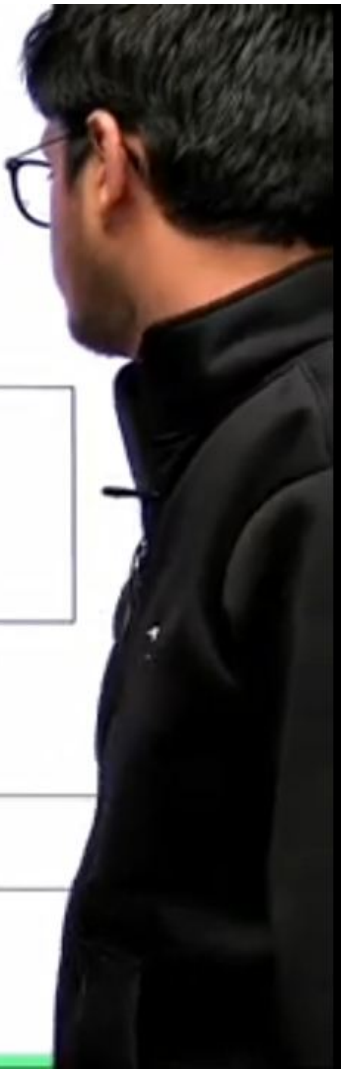
Problem 9.a

```
class B{
public:
    virtual void fun1(){
        cout<<"Base ";
    }
    void fun2(){
        cout<<"Base ";
    }
};

class D: public B{
public:
    void fun1(){
        cout<<"Derived ";
    }
    void fun2(){
        cout <<"Derived ";
    }
};
```

Output ?

```
B *ptr;
D obj;
ptr = &obj;
ptr->fun2();
```

Base



Problem 9.b

```
class B{
public:
    virtual void fun1(){
        cout<<"Base ";
    }
    void fun2(){
        cout<<"Base ";
    }
};

class D: public B{
public:
    void fun1(){
        cout<<"Derived ";
    }
    void fun2(){
        cout <<"Derived ";
    }
};
```

Output ?

```
B *ptr;
D obj;
ptr = &obj;
ptr->fun1();
```

Derived

Problem 9.c

```
class B{
public:
    virtual void fun1(){
        cout<<"Base ";
    }
    void fun2(){
        cout<<"Base ";
    }
};

class D: public B{
public:
    void fun2(){
        cout <<"Derived ";
    }
};
int main()
{
    B *ptr;
    D obj;
    ptr = &obj;
    ptr->fun1();
}
```

Output ?

Base

Problem 10

```
class B{
public:
    virtual void fun1() = 0;
    void fun2(){
        cout<<"I am Base"<<endl;
    }
};
class D: public B{
public:
    void fun1(){
        cout <<"Implemented pure virtual function"<<endl;
    }
};
int main(){
    B ptr;
    D obj;
    obj.fun2();
}
```

Output ?

Error, we can only have pointers and references of abstract class type

Problem 11

```
class B{
public:
    virtual void fun1() = 0;
    void fun2(){
        cout<<"I am Base"<<endl;
    }
};
class D: public B{
public:
    void fun1(){
        cout <<"Implemented pure virtua
    }
};
int main(){
    B *ptr;
    D obj;
    ptr = &obj;
    ptr->fun2();
}
```

Output ?

I am Base

Problem 12.a

```
class A{
public:
    virtual void fun1() = 0;
    void fun2(){
        cout<<"I am Base"<<endl;
    }
};
class B: public A{
public:
    void fun1(){
        cout <<"Implemented PVF A"<<endl;
    }
    virtual void fun3() = 0;
};
class C: public B{
public:
    void fun3(){
        cout<<"Implemented PVF B"<<endl;
    }
};
```

Output ?

```
C obj;
obj.fun2();
```

I am Base

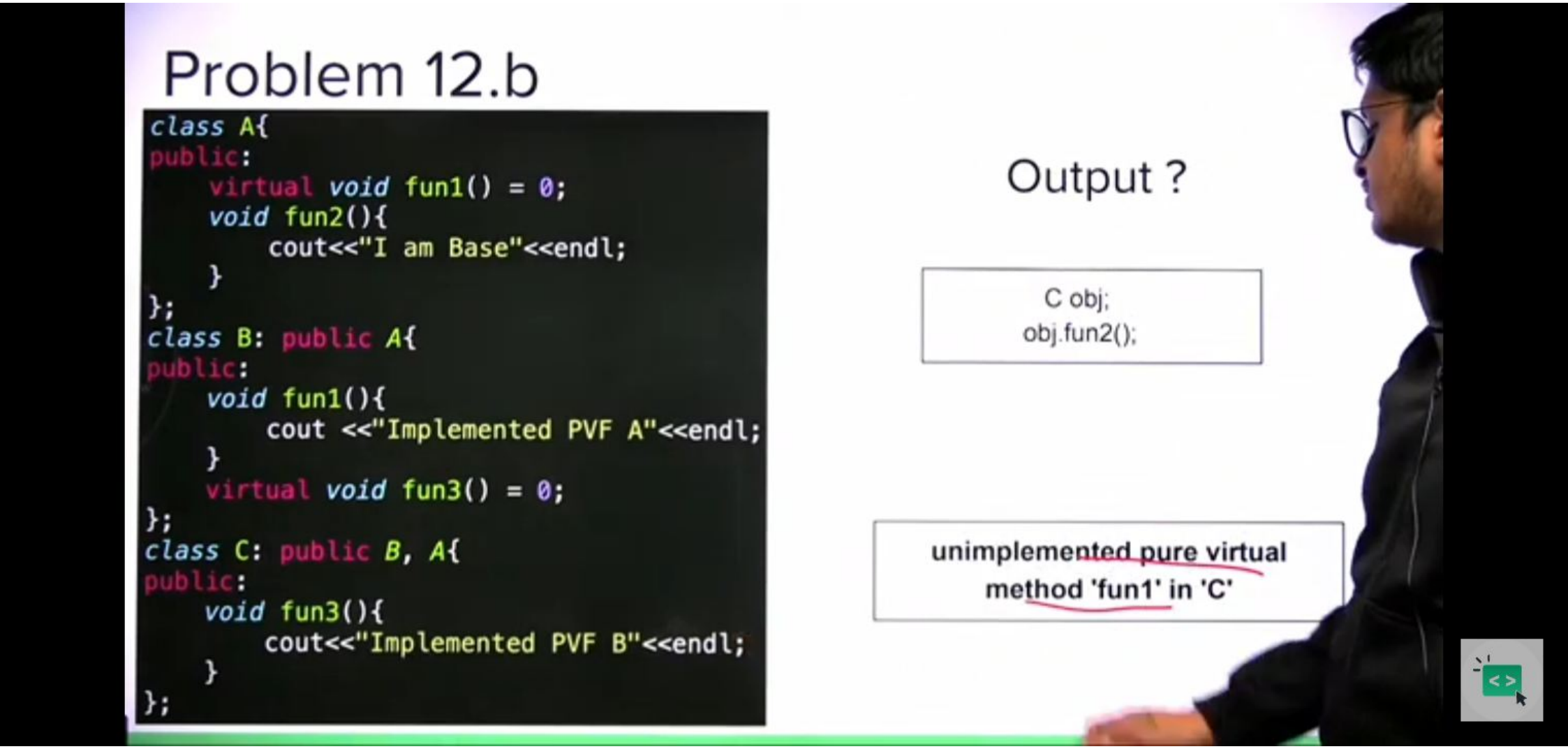
Problem 12.b

```
class A{
public:
    virtual void fun1() = 0;
    void fun2(){
        cout<<"I am Base"<<endl;
    }
};
class B: public A{
public:
    void fun1(){
        cout <<"Implemented PVF A"<<endl;
    }
    virtual void fun3() = 0;
};
class C: public B, A{
public:
    void fun3(){
        cout<<"Implemented PVF B"<<endl;
    }
};
```

Output ?

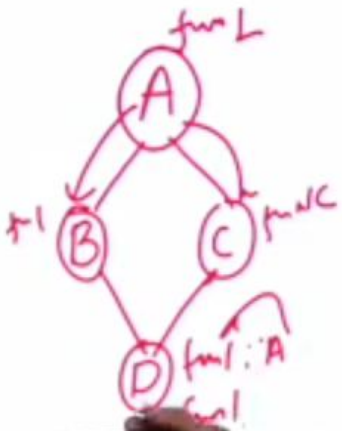
```
C obj;
obj.fun2();
```

```
unimplemented pure virtual
method 'fun1' in 'C'
```



Problem 13

```
class A{
public:
    virtual void fun1(){
        cout<<"A"<<endl;
    }
};
class B: public A{ };
class C: public A{ };
class D: public B, C{
public:
    void fun2(){
        fun1();
        cout<<"D class"<<endl;
    }
};
int main(){
    D obj;
    obj.fun1();
}
```

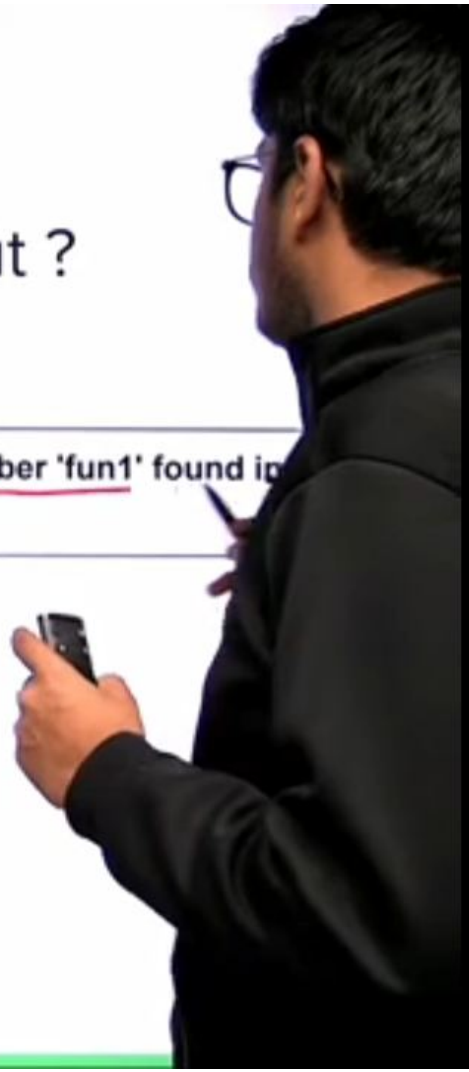


Problem 13

```
class A{
public:
    virtual void fun1(){
        cout<<"A"<<endl;
    }
};
class B: public A{ };
class C: public A{ };
class D: public B, C{
public:
    void fun2(){
        fun1();
        cout<<"D class"<<endl;
    }
};
int main(){
    D obj;
    obj.fun1();
}
```

Output ?

error: non-static member 'fun1' found in
multiple base-class



Problem 14.a

```
class A{
public:
    virtual void fun1(){
        cout<<"A"<<endl;
    }
};
class B: virtual public A{ };
class C: virtual public A{ };
class D: public B, C{
public:
    void fun2(){
        fun1();
        cout<<"D class"<<endl;
    }
};
int main(){
    D obj;
    obj.fun1();
}
```

Output ?

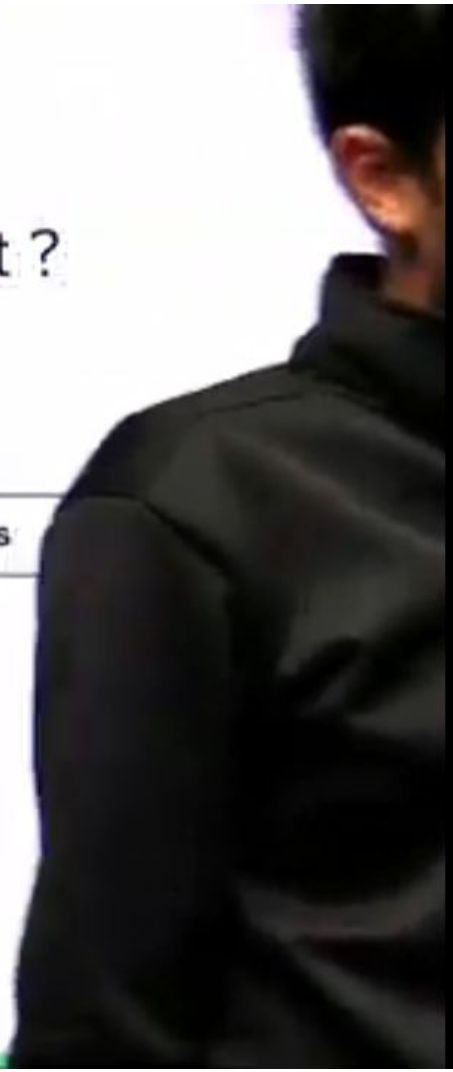
A

Problem 14.b

```
class A{
public:
    virtual void fun1(){
        cout<<"A"<<endl;
    }
};
class B:  public A{ };
class C:  public A{ };
class D: public B, C{
public:
    void fun2(){
        cout<<"D class"<<endl;
    }
};
int main(){
    D obj;
    obj.fun2();
}
```

Output ?

D class



Problem 15.a

```
class A{
public:
    virtual void fun1(){
        cout<<"A";
    }
};
class B: public A{
public:
    void fun1(){
        cout<<"B";
    }
};
class C: public B{
public:
    void fun1(){
        cout<<"C";
    }
};
```

Output ?

```
A *ptr;
B b; C c;
ptr = &b;
ptr->fun1();
ptr = &c;
ptr->fun1();
```

BC

Problem 15.b

```
class A{
public:
    virtual void fun1(){
        cout<<"A";
    }
};
class B: public A{
public:
    virtual void fun1(){
        cout<<"B";
    }
};
class C: public B{ };
```

Output ?

```
A *ptr;
B b; C c;
ptr = &b;
ptr->fun1();
ptr = &c;
ptr->fun1();
```

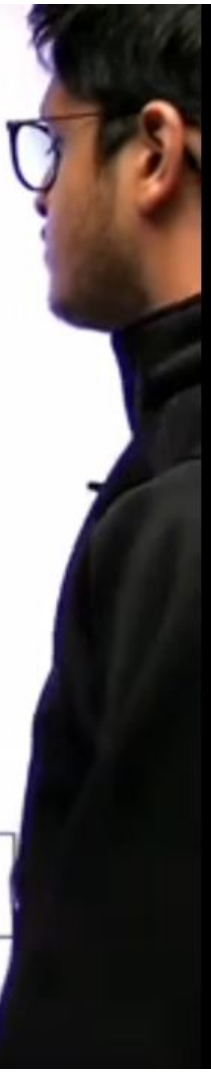
Problem 15.b

```
class A{
public:
    virtual void fun1(){
        cout<<"A";
    }
};
class B: public A{
public:
    virtual void fun1(){
        cout<<"B";
    }
};
class C: public B{ };
```

Output ?

```
A *ptr;
B b; C c;
ptr = &b;
ptr->fun1();
ptr = &c;
ptr->fun1();
```

BB



Problem 1

Which of the following is not a feature of OOP

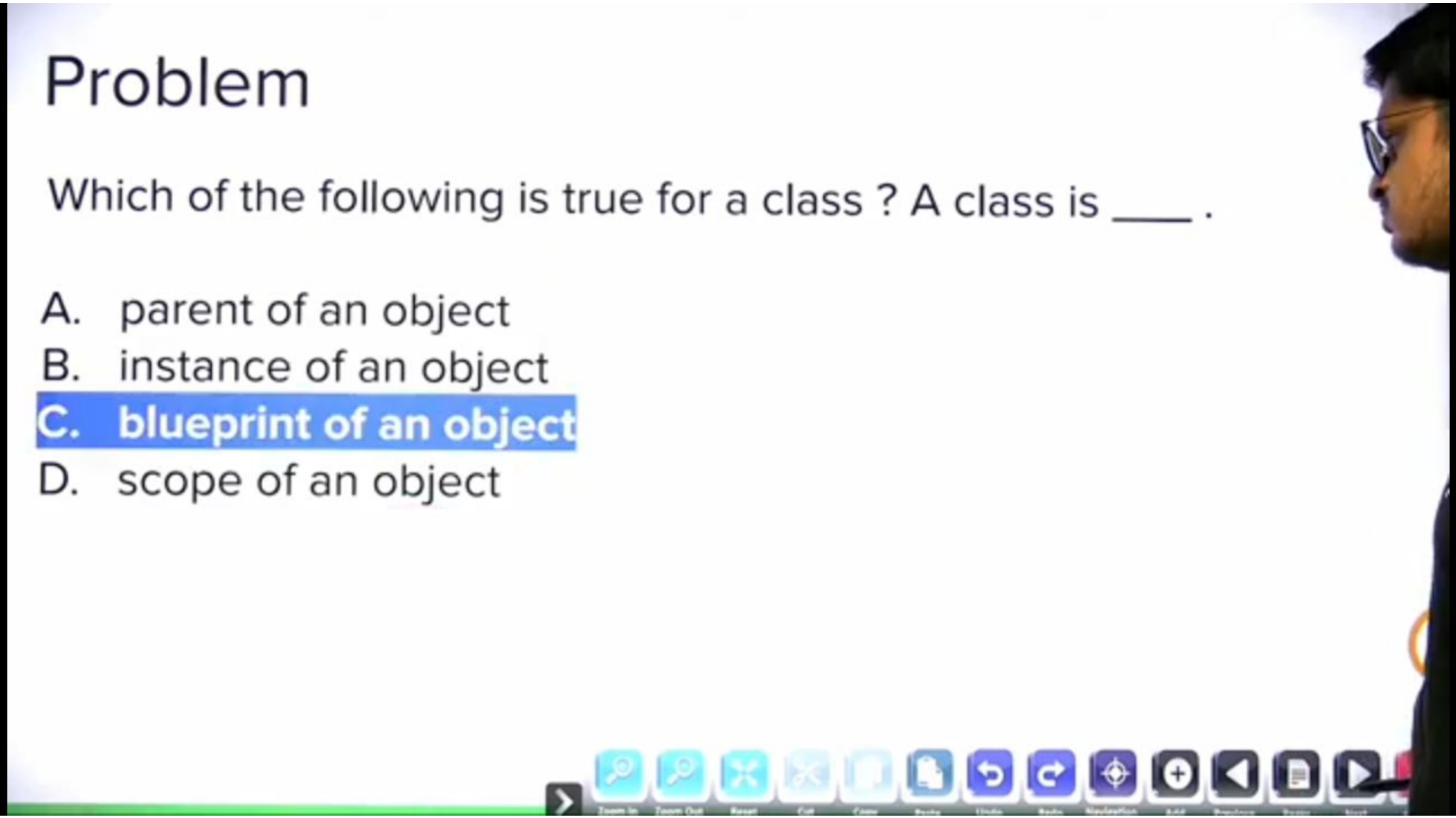
A) Encapsulation
B) Pointers
C) Data hiding
D) Inheritance

Abstraction
Polymo

Problem

Which of the following is true for a class ? A class is ____ .

A. parent of an object
B. instance of an object
C. **blueprint of an object**
D. scope of an object



Problem

Which class cannot create its instance?

A. Virtual class
B. Nested class
C. Derived virtual class
D. **Abstract class**

Problem

Which of the following is not considered as a member of class?

A. pure virtual function
B. const function
C. **friend function**
D. static function

Problem

```
class One{
public:
    virtual int fun(int x, int y){

        return(x++ + ++y + x<<2);
    }
};
class Two: public One{
public:
    int fun(int x, int y){
        return(++x + y++ + y>>2);
    }
};
int main(){
    One obj = Two();
    cout<<obj.fun(2, 3)<<endl;
}
```

Output?

Problem

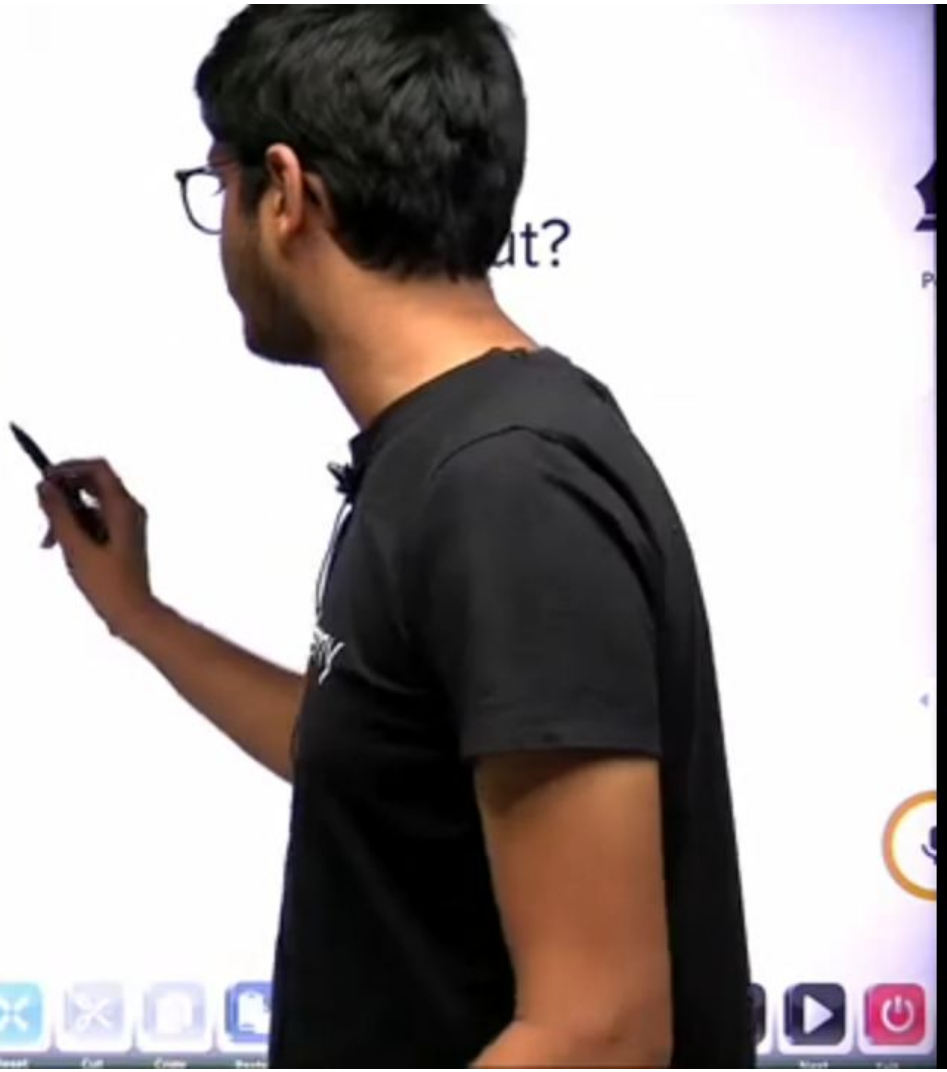
```
class One{
public:
    virtual int fun(int x, int y){

        return(x++ + ++y + x<<2);
    }
};
class Two: public One{
public:
    int fun(int x, int y){
        return(++x + y++ + y>>2);
    }
};
int main(){
    One obj = Two();
    cout<<obj.fun(2, 3)<<endl;
}
```

Output?

36

Problem

```
class One{
public:
    virtual int fun(int x, int y){
        return(x++ + ++y + x<<2);
    }
};
class Two: public One{
public:
    int fun(int x, int y){
        return(++x + y++ + y>>2);
    }
};
int main(){
    One *ptr;
    Two obj;
    ptr = &obj;
    cout<<ptr->fun(2, 3)<<endl;
}
```



Problem

```
class One{
public:
    virtual int fun(int x, int y){
        return(x++ + ++y + x<<2);
    }
};
class Two: public One{
public:
    int fun(int x, int y){
        return(++x + y++ + y>>2);
    }
};
int main(){
    One *ptr;
    Two obj;
    ptr = &obj;
    cout<<ptr->fun(2, 3)<<endl;
}
```

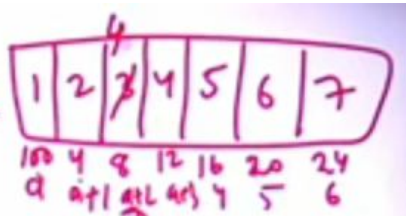
Output?

2

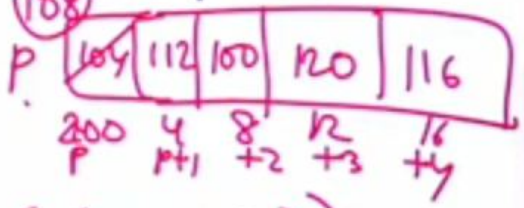
Problem

```
#include<iostream>
using namespace std;
int main(){
    int a[] = {1, 2, 3, 4, 5, 6, 7};
    int *p[] = {a+1, a+3, a, a+5, a+4};
    int **q = p;
    cout<<*a<<" "<<**p<<" "<<**q<<endl;
    ++*++*q;
    cout<<*a<<" "<<**p<<" "<<**q<<endl;
    ++++*p;
    ++++++q;
    cout<<*a<<" "<<**p<<" "<<**q<<endl;
    return 0;
}
```

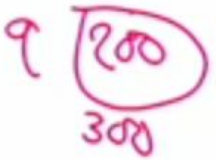
*, ++ Same
Right to left



Output ?



++(*(++(*q)))



Problem

```
#include<iostream>
using namespace std;
int main(){
    int a[] = {1, 2, 3, 4, 5, 6, 7};
    int *p[] = {a+1, a+3, a, a+5, a+4};
    int **q = p;
    cout<<*a<<" "<<**p<<" "<<**q<<endl;
    ++*++*q;
    cout<<*a<<" "<<**p<<" "<<**q<<endl;
    ++++*p;
    ++*++++q;
    cout<<*a<<" "<<**p<<" "<<**q<<endl;
    return 0;
}
```

Output ?

1 2 2
1 4 4
1 5 2

Problem

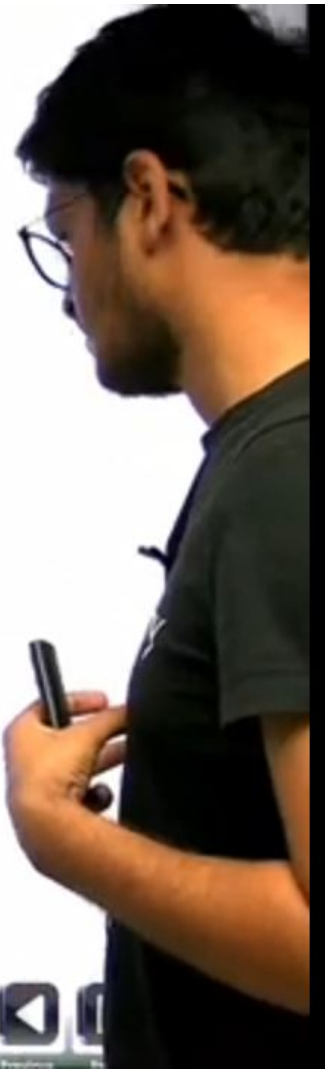
What members of the base are never accessible to the child?

A. **Private members**
B. Protected members
C. Both Private and Protected members
D. None of the above

Problem

A class is tagged abstract, if it has ____ .

A. Exactly one virtual method
B. Atleast one virtual method
C. Exactly one pure virtual method
D. **Atleast one pure virtual method**

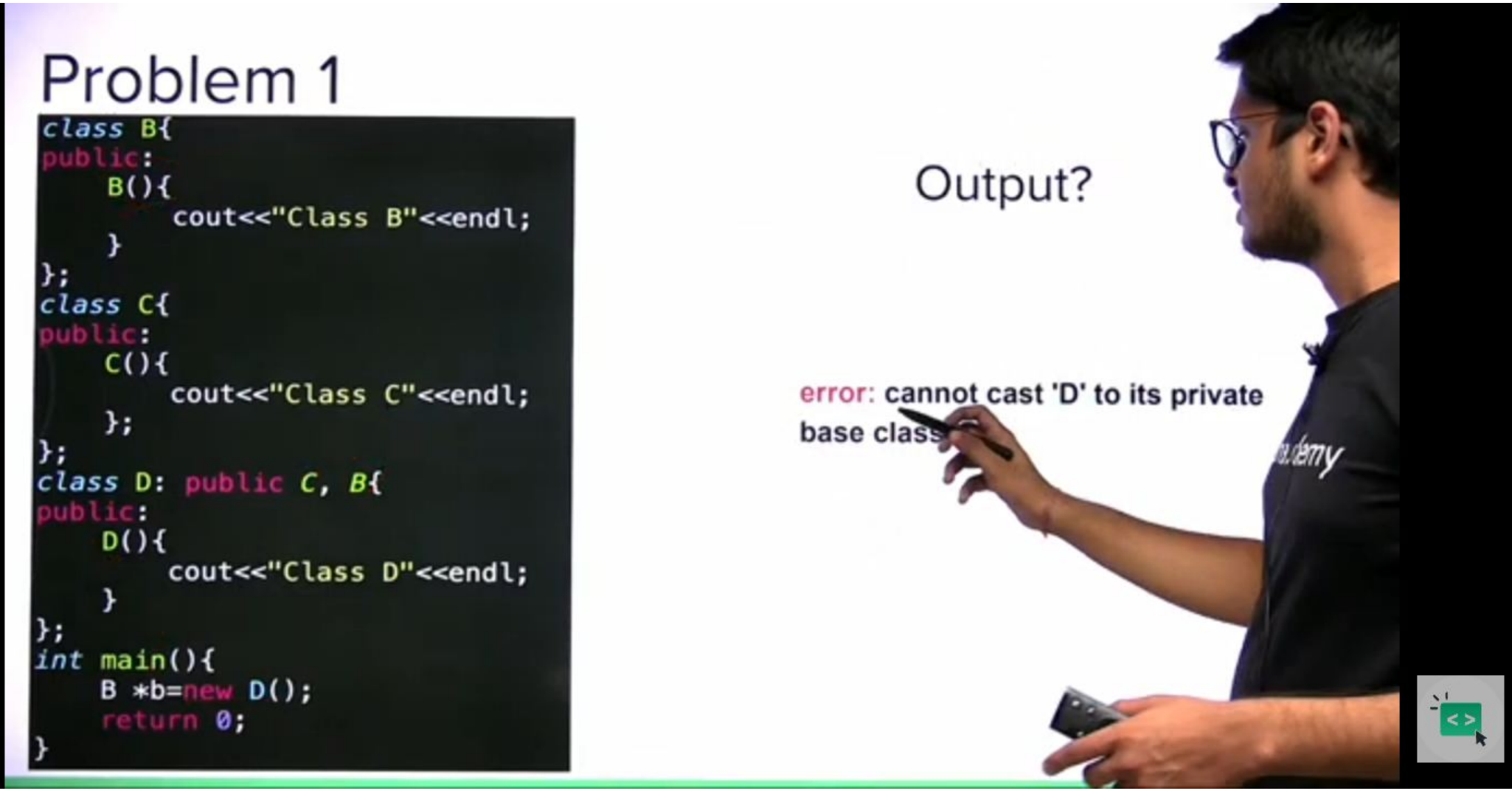


Problem 1

```
class B{
public:
    B(){
        cout<<"Class B"<<endl;
    }
};
class C{
public:
    C(){
        cout<<"Class C"<<endl;
    };
};
class D: public C, B{
public:
    D(){
        cout<<"Class D"<<endl;
    }
};
int main(){
    B *b=new D();
    return 0;
}
```

Output?

error: cannot cast 'D' to its private base class

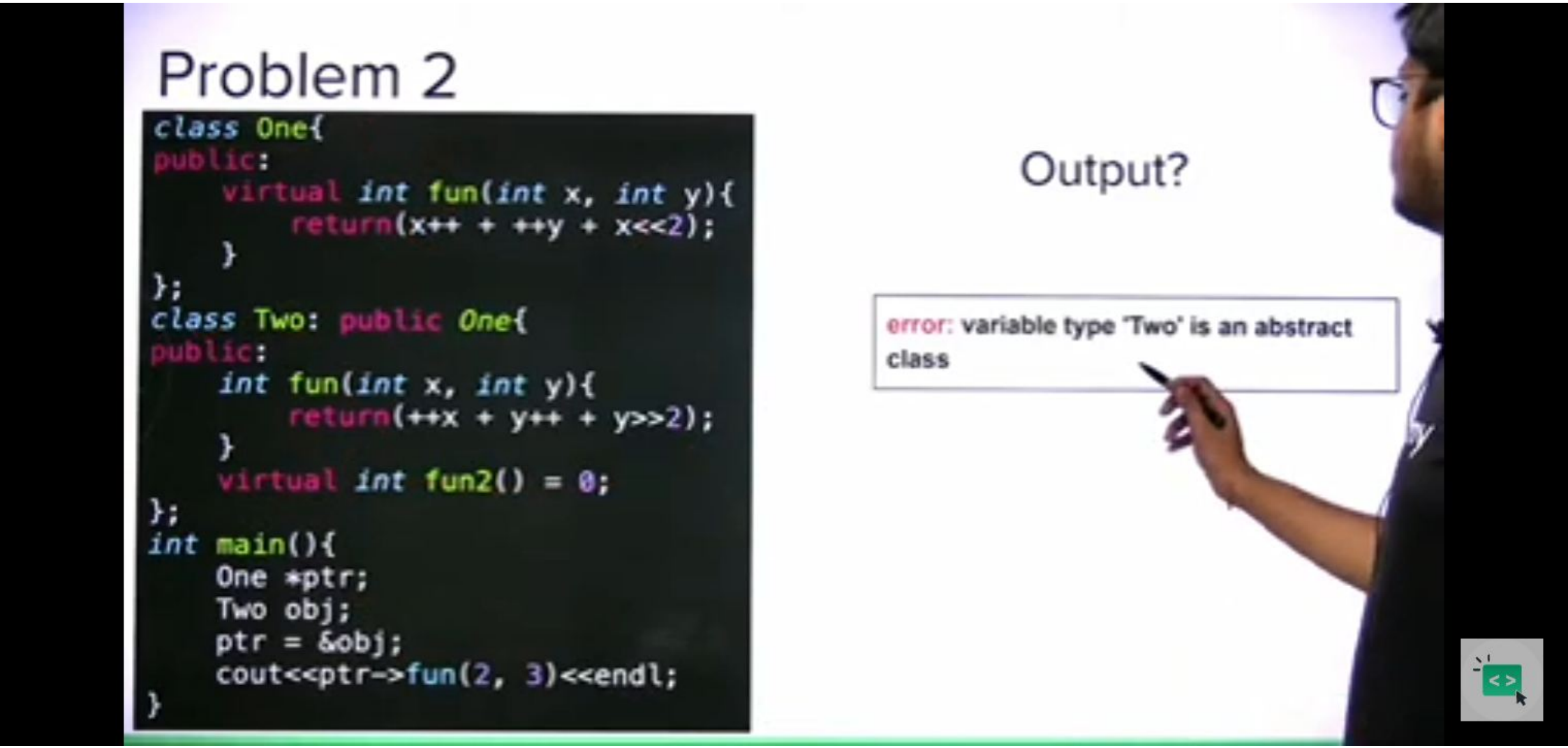


Problem 2

```
class One{
public:
    virtual int fun(int x, int y){
        return(x++ + ++y + x<<2);
    }
};
class Two: public One{
public:
    int fun(int x, int y){
        return(++x + y++ + y>>2);
    }
    virtual int fun2() = 0;
};
int main(){
    One *ptr;
    Two obj;
    ptr = &obj;
    cout<<ptr->fun(2, 3)<<endl;
}
```

Output?

error: variable type 'Two' is an abstract class

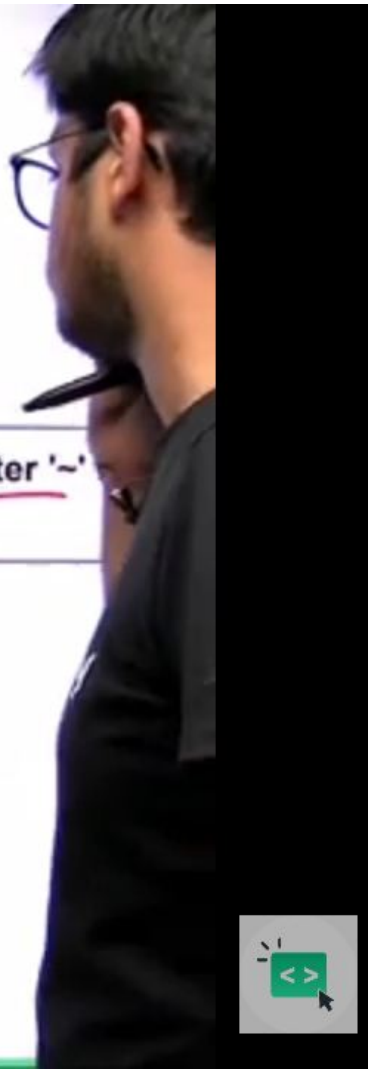


Problem 4

```
class B{
public:
    virtual ~B()=0;
};
class D : public B{
public:
    ~D(){
        cout<<"Derived destructor"<<endl;
    }
    ~B(){
        cout<<"Implemented PVF"<<endl;
    }
};
int main(){
    B *b=new D();
    delete b;
    return 0;
}
```

Output?

error: expected the class name after '~' to name the enclosing class

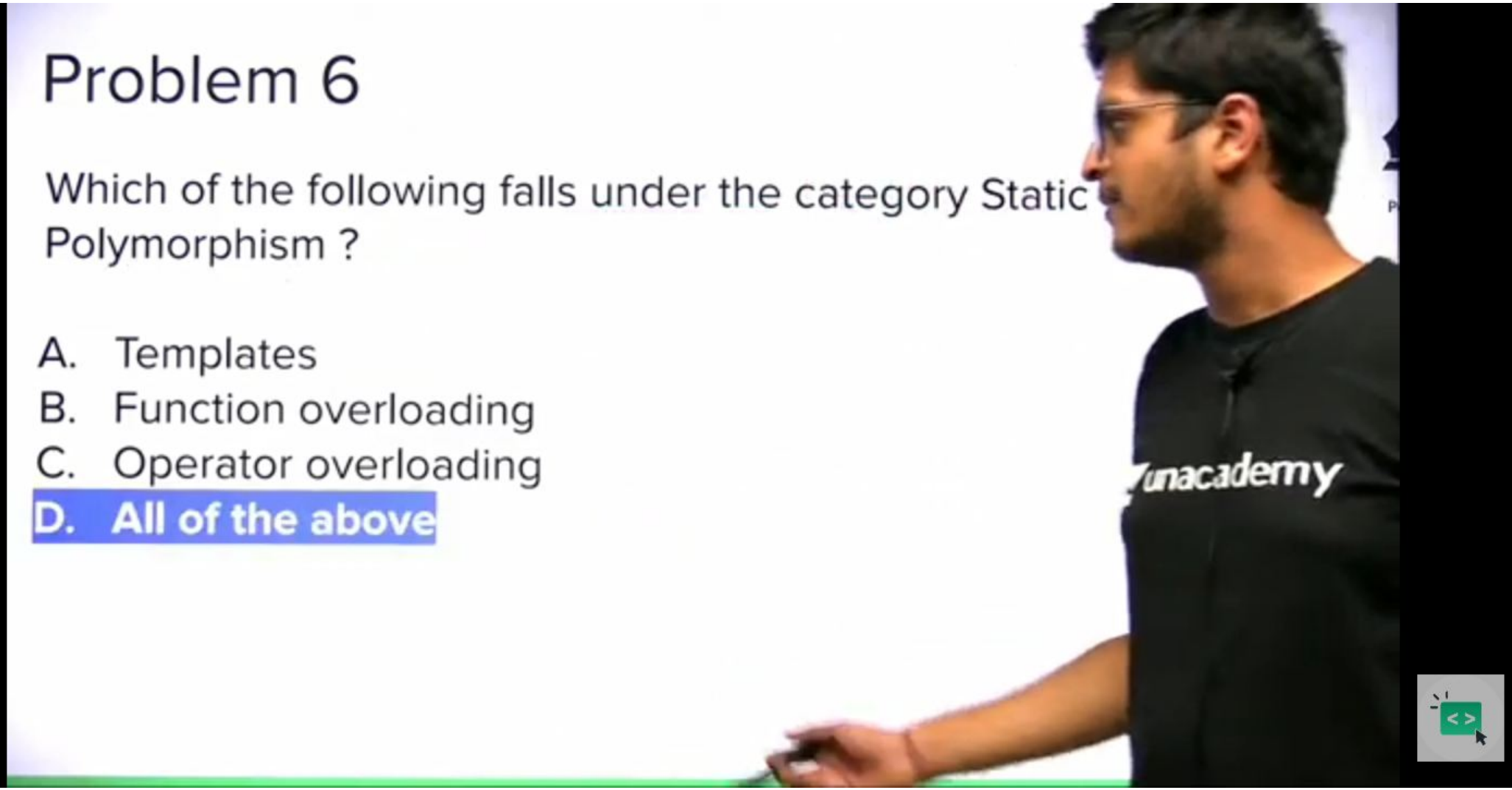


Problem 5

```
class B{
public:
    virtual ~B()=0;
};
B::~B(){
        cout<<"Implemented PVF"<<endl;
    }
class D : public B{
public:
    ~D(){
        cout<<"Derived destructor"<<endl;
    }
};
int main(){
    B *b=new D();
    delete b;
    return 0;
}
```

Output?

Derived destructor
Implemented PVF


Problem 6
Which of the following falls under the category Static Polymorphism ?
A. Templates
B. Function overloading
C. Operator overloading
D. All of the above
unacademy

Problem 7

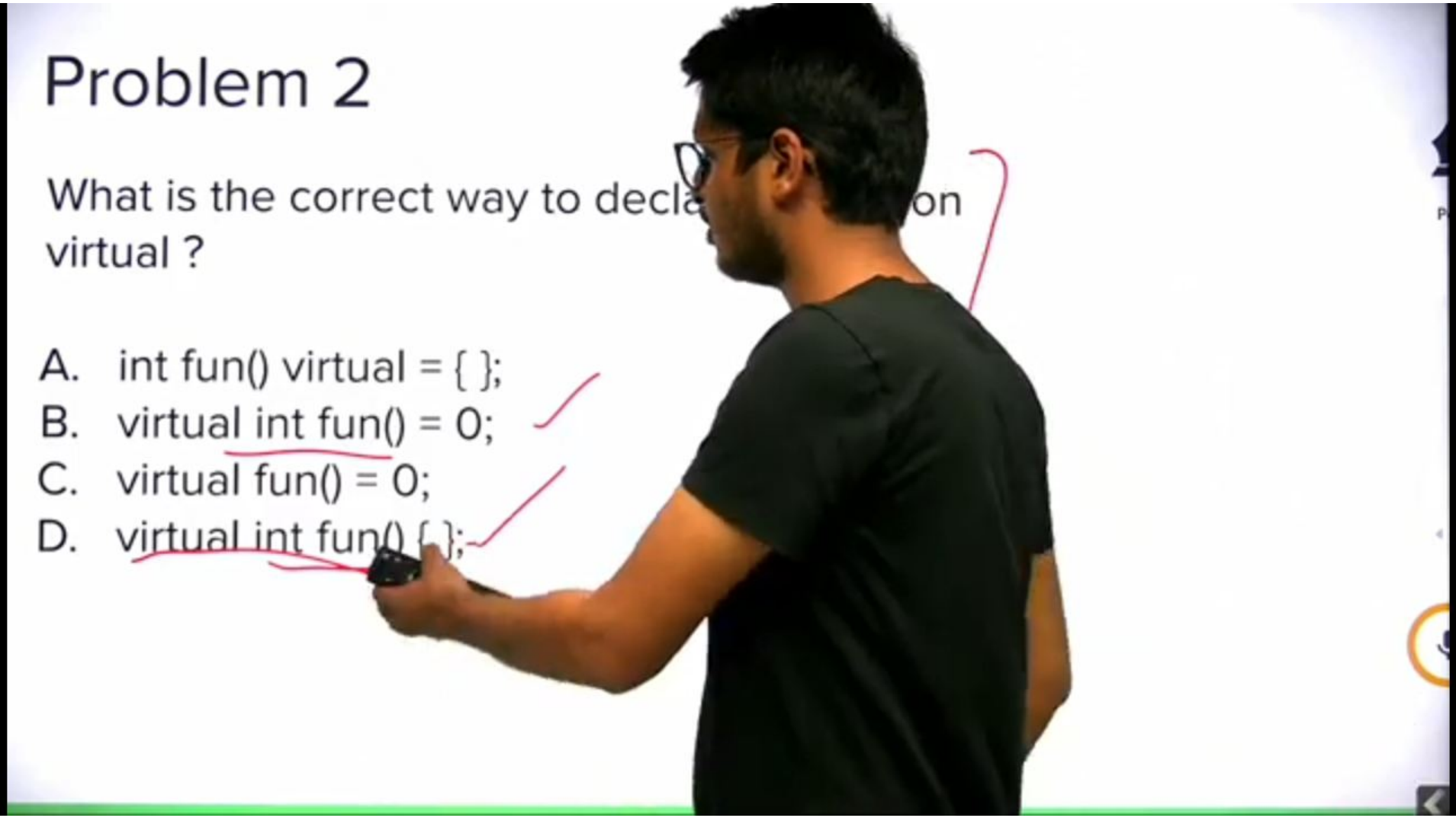
What role a constructor plays in the class ?

A. Constructs a new class
B. Constructs a new object
C. Constructs a new function
D. **Initializes an object**

Problem 1

Which of the following is not a type of constructor?

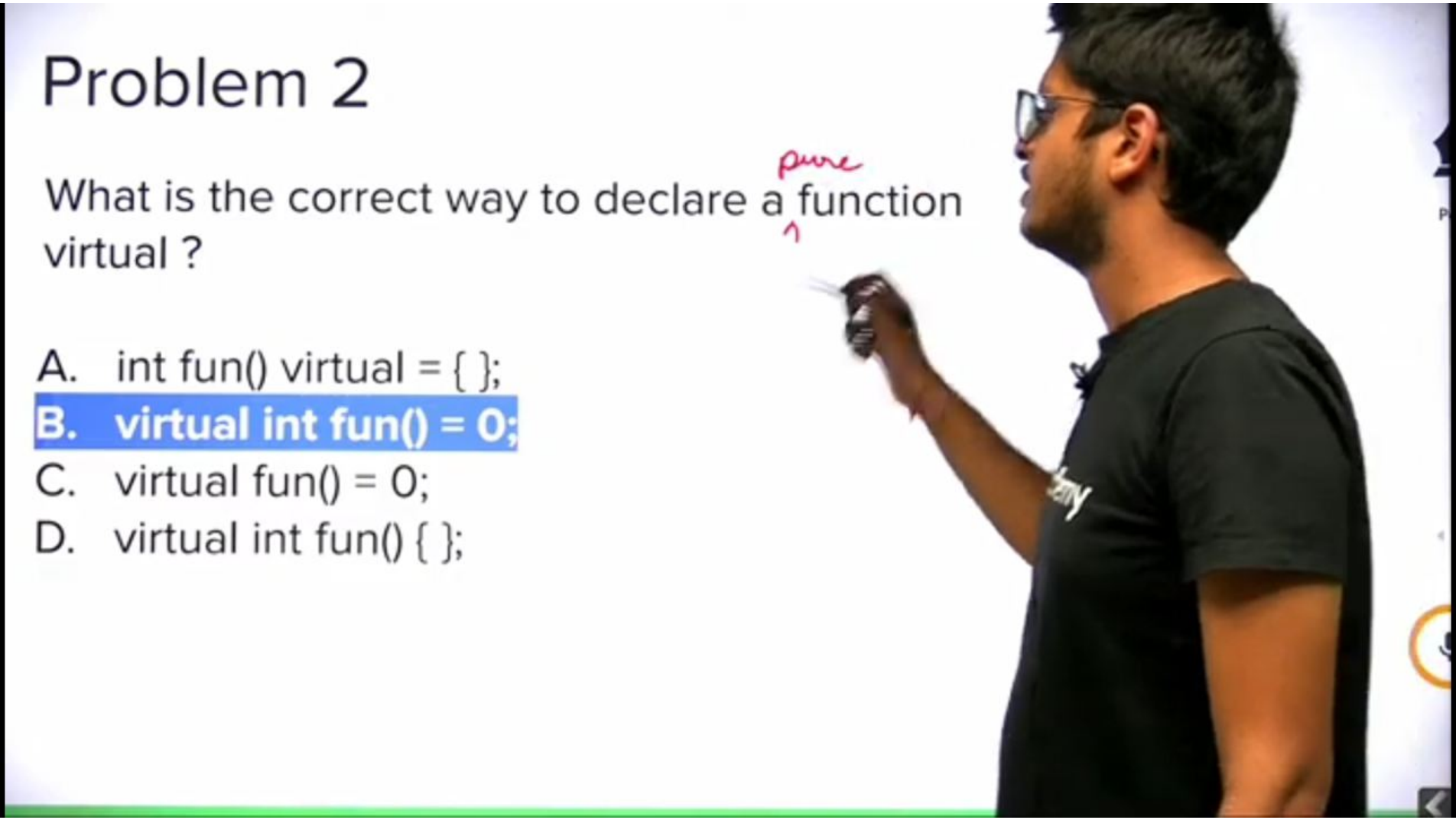
A. Copy constructor
B. Friend constructor
C. Parameterized constructor
D. Default constructor


Problem 2
What is the correct way to decla on virtual ?
A. int fun() virtual = { };
B. virtual int fun() = 0;
C. virtual fun() = 0;
D. virtual int fun() { };

Problem 2

What is the correct way to declare a pure function virtual ?

A. int fun() virtual = { };
B. **virtual int fun() = 0;**
C. virtual fun() = 0;
D. virtual int fun() { };



Problem 3

Which of the following is/are allowed in C++, B being base class and D being derived class

A. B obj = D();

B. D obj = B();

C. B *ptr; D obj;
ptr = &obj;

D. D *ptr; B obj;
ptr = &obj;

Problem 4

```
class A{
    int a, b;
public:
    A(int x, int y){
        a = x;
        b = y;
    }
    void swap(A ob){
        int temp = ob.a;
        ob.a = a; a = temp;
        temp = ob.b;
        ob.b = b; b = temp;
    }
    void print(){
        cout<<a<<b<<" ";
    }
};
int main(){
    A obj1(4, 5), obj2(2, 6);
    obj1.print();
    obj1.swap(obj2);
    obj1.print();
}
```

Output?

45 26

Problem 5

```
class A{
    int a, b;
public:
    A(int x, int y){
        a = x;
        b = y;
    }
    void swap(A ob){
        int temp = ob.a;
        ob.a = a; a = temp;
        temp = ob.b;
        ob.b = b; b = temp;
    }
    void print(){
        cout<<a<<b<<" ";
    }
};
int main(){
    A obj1(4, 5), obj2(2, 6);
    obj2.print();
    obj1.swap(obj2);
    obj2.print();
}
```

Output?

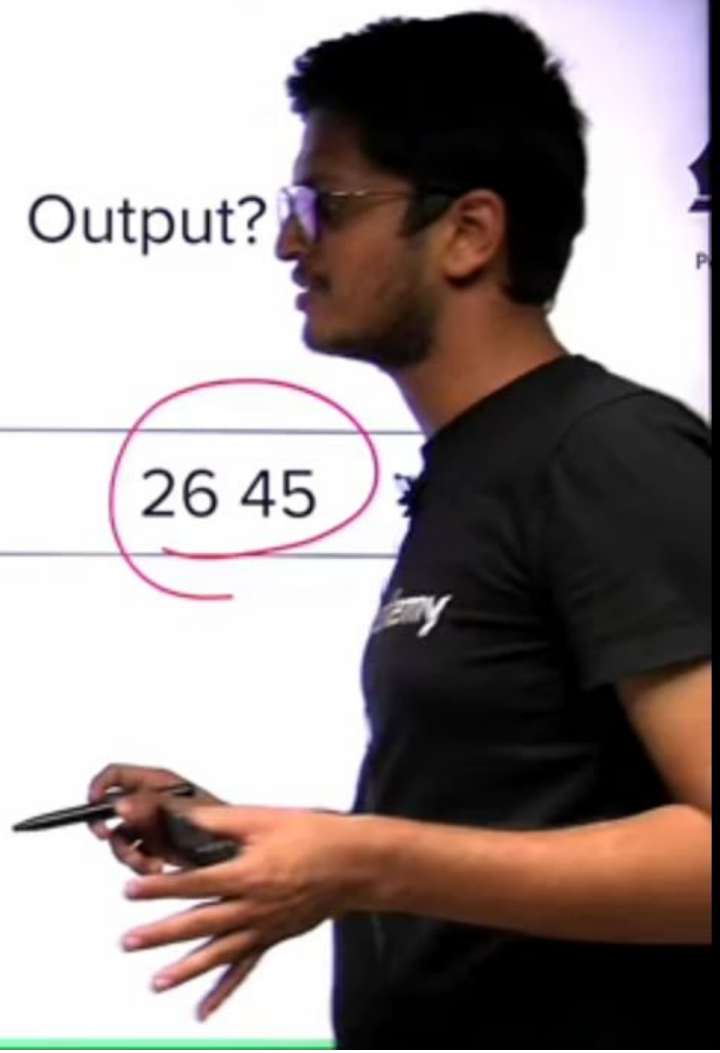
26 26

Problem 6

```
class A{
    int a, b;
public:
    A(int x, int y){
        a = x;
        b = y;
    }
    void swap(A &ob){
        int temp = ob.a;
        ob.a = a; a = temp;
        temp = ob.b;
        ob.b = b; b = temp;
    }
    void print(){
        cout<<a<<b<<" ";
    }
};
int main(){
    A obj1(4, 5), obj2(2, 6);
    obj2.print();
    obj1.swap(obj2);
    obj2.print();
}
```

Output?

26 45



Problem 7

```cpp
#include<iostream>
using namespace std;
long long fun(long long n){
    long long result = 0;
    while(n!=0){
        result = result*10 + n%10;
        n /= 10;
    }
    return result;
}
int main(){
    long long n;
    cin>>n;
    cout<<fun(n)<<endl;
    return 0;
}
```

Calculate the output on input n = 163378938293345

Output?

54392839873361

Problem 8

```
class A{
    int a;
public:
    A(int x) { a= x; }
    int fun(){
        return a*a;
    }
    int fun(int x){
        return x*x;
    }
    void fun(int x, int y){
        cout<<(x*y)<<endl;
    }
};
class B: public A{
    int a, b;
public:
    B(int x, int y): A(y){
        a = x;
        b = y;
    }
};
```

Output?

```
B obj1(1, 4);
cout<<obj1.fun();
obj1.fun(1, 4);
```

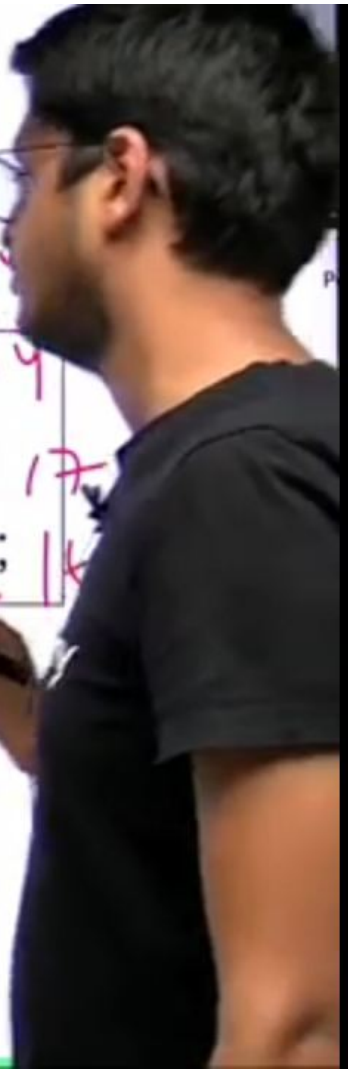
164

Problem 9

```
class A{
    int a;
public:
    A(int x) { a= x; }
    int fun(){
        return a*a;
    }
    void fun(int x, int y, int z){
        cout<<(x*y*z)<<endl;
    }
};
class B: public A{
    int a, b;
public:
    B(int x, int y): A(y){
        a = x;
        b = y;
    }
    int fun(){
        return a*a+b*b;
    }
};
```

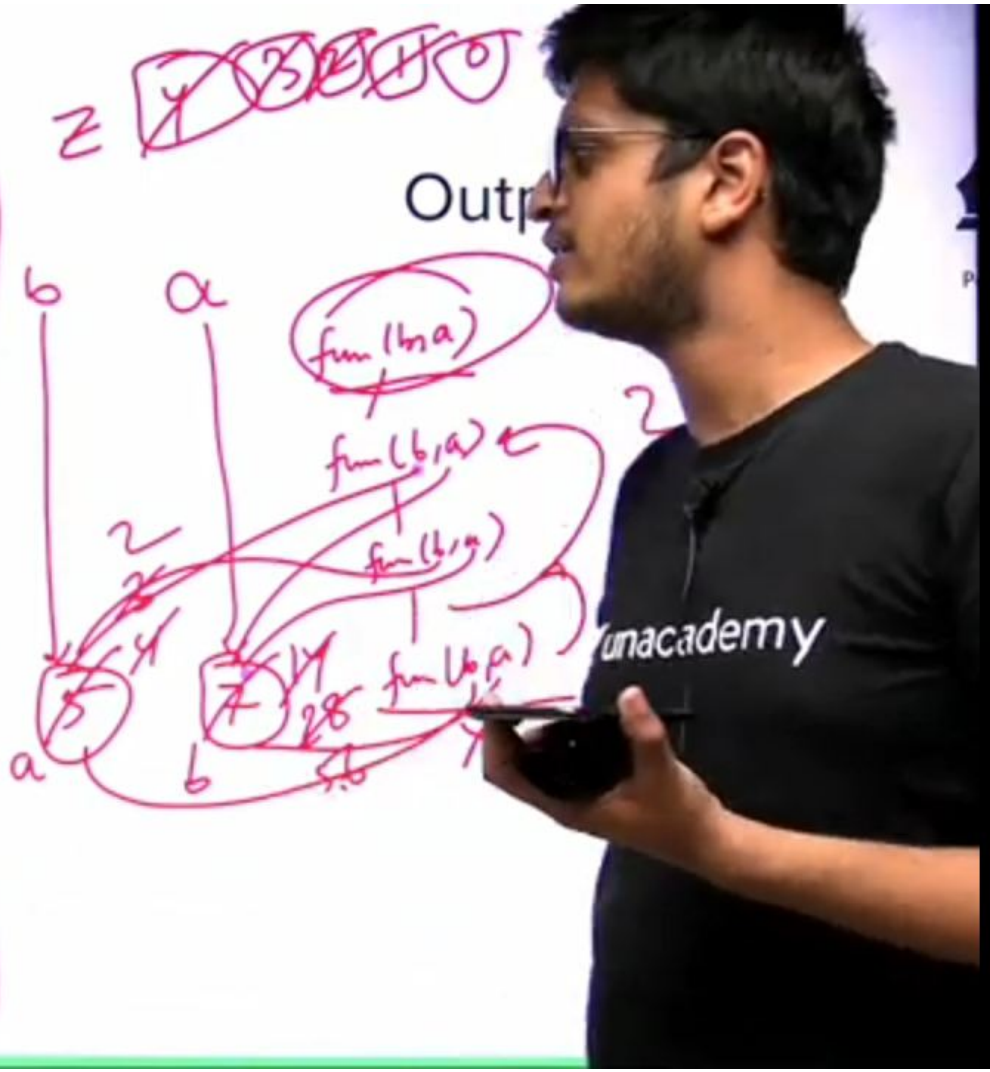
Output?

```
B obj1(1, 4);
cout<<obj1.fun();
obj1.A::fun(1, 4, 4);
```



Problem 0

```
#include<iostream>
using namespace std;
int fun(int &b, int &a){
    static int z=4;
    while(--z>0){
        fun(b, a);
        a*=2;
        b--;
    }
    return a+b+10;
}
int main(){
    int a=5, b=7;
    cout<<fun(a, b)<<endl;
    cout<<a<<" "<<b<<endl;
    return 0;
}
```

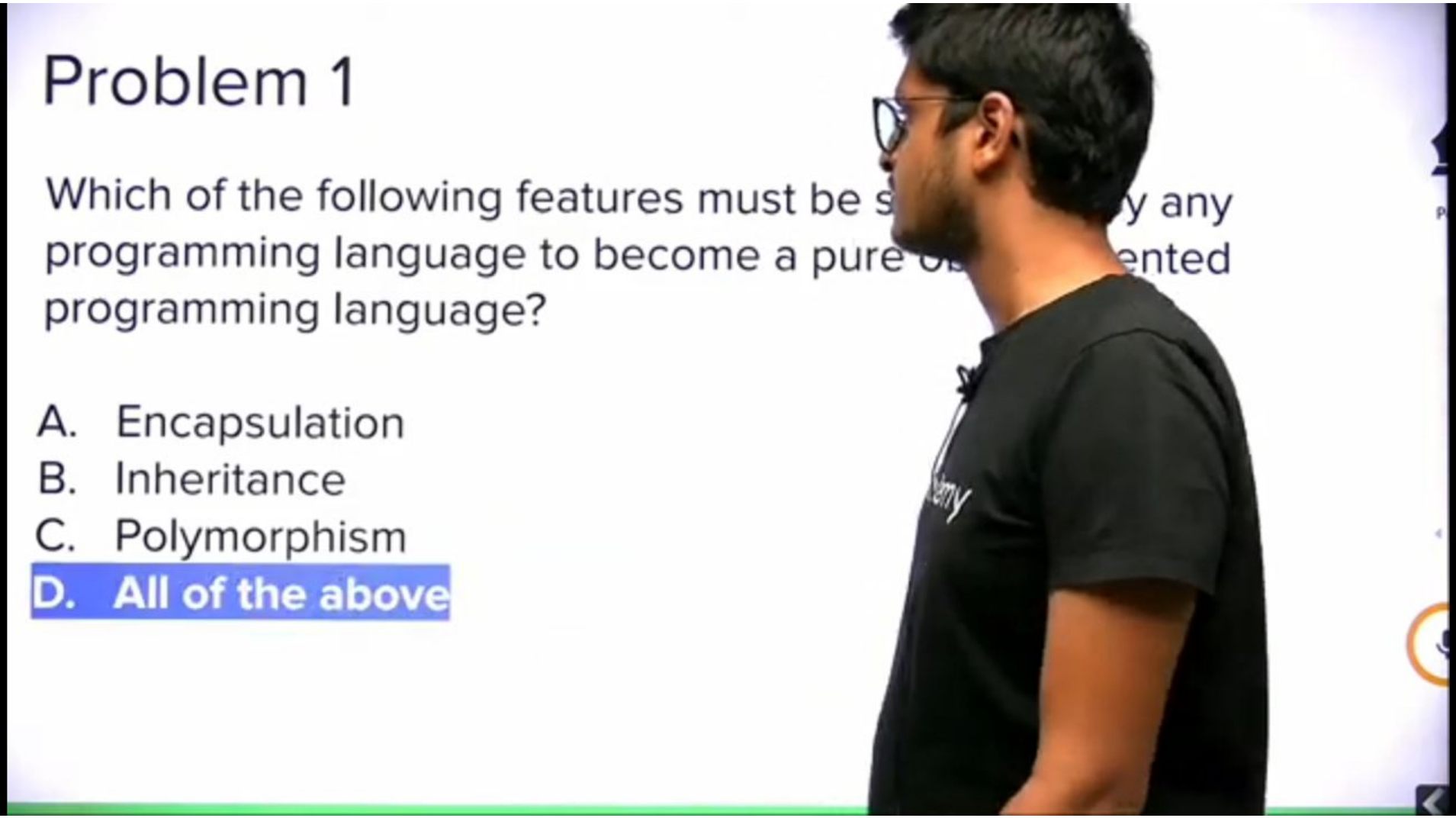


Problem 0

```
#include<iostream>
using namespace std;
int fun(int &b, int &a){
    static int z=4;
    while(--z>0){
        fun(b, a);
        a*=2;
        b--;
    }
    return a+b+10;
}
int main(){
    int a=5, b=7;
    cout<<fun(a, b)<<endl;
    cout<<a<<" "<<b<<endl;
    return 0;
}
```

Output

68
2 56

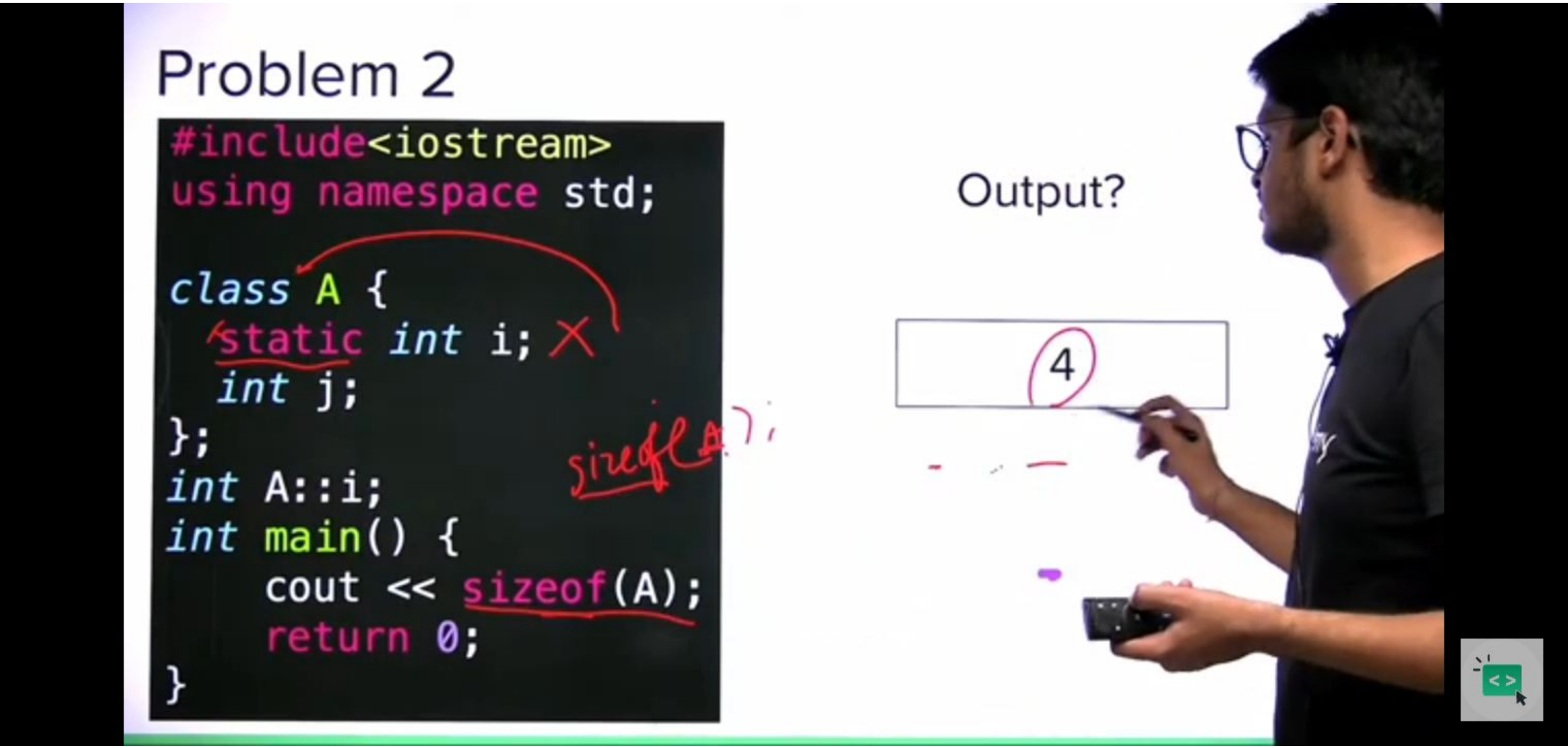

Problem 1
Which of the following features must be s y any
programming language to become a pure o ented
programming language?
A. Encapsulation
B. Inheritance
C. Polymorphism
D. All of the above

Problem 2

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

class A {
  static int i;
   int j;
};
int A::i;
int main() {
    cout << sizeof(A);
    return 0;
}
```

Output?

4

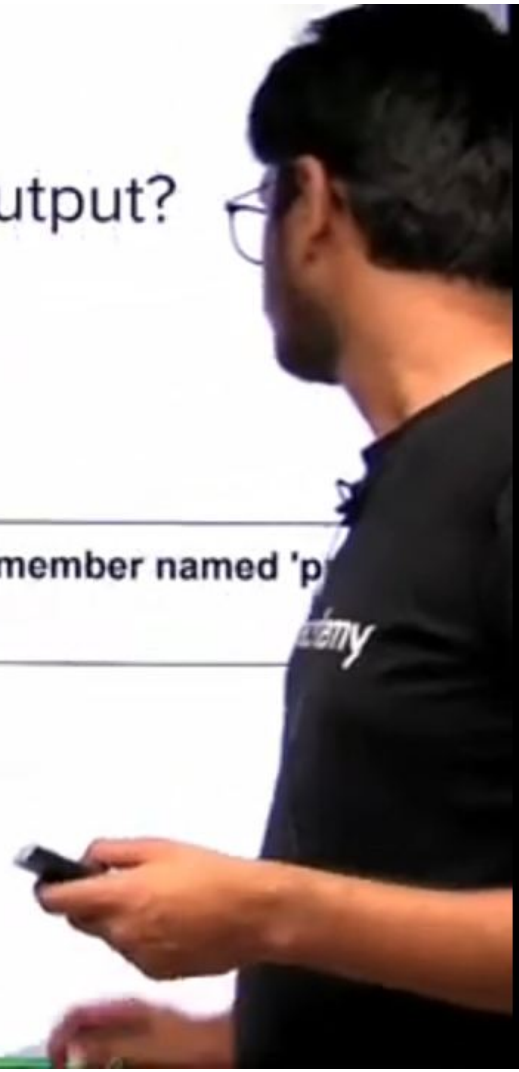


Problem 4

```
class Host{
public:
  class Nested{
  public:
    void print(){
      cout << "Hello world"<<endl;
    }
  };
};
int main(){
  Host bar;
  bar.print();
  return 0;
}
```

Output?

error: no member named 'p
in 'Host'



Problem 5

```
class Host{
public:
  class Nested{
  public:
    void print(){
      cout << "Hello world"<<endl;
    }
  };
};
int main(){
  Host::Nested foo;
  foo.print();
  return 0;
}
```

Output?

Hello world

Problem 6

```
class Host{
    int a;
public:
  class Nested{
    Host h;
  public:
    Nested(){
        h.a = 15;
    }
    int get(){
        return h.a;
    }
  };
};
int main(){
  Host::Nested foo;
  cout<<foo.get();
  return 0;
}
```

Output?

error: field has incomplete type 'Host'
(definition of 'Host' is not complete until the closing '}')

Problem 7

```
#include<iostream>
using namespace std;
void gun(string s){
    if(s[0]=='\0') return;
    gun(s.substr(1));
    cout<<s[0];
}
int main(){
    string s;
    cin>>s;
    gun(s);
    cout<<endl;
    return 0;
}
```

Calculate the output on s = "abac" (without quotes)

Output?

caba