

50 Practice Problems in C++ OOPs - Part I | L:29 | C++

Q1:

Problem 1

Which of the following is not a feature of OOPs:

- A) Encapsulation ✓
- B) Pointers ↗
- C) Data hiding ✓
- D) Inheritance ✓

Abstraction
Polymorphism



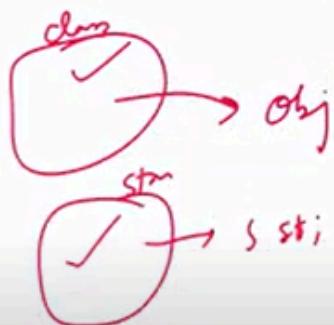
Encapsulation, data hiding, inheritance, abstraction, polymorphism are the features of oops.
pointers ==> not the features of oops

Q2:

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 ✗ *mine*



Class is the blueprint of the object.

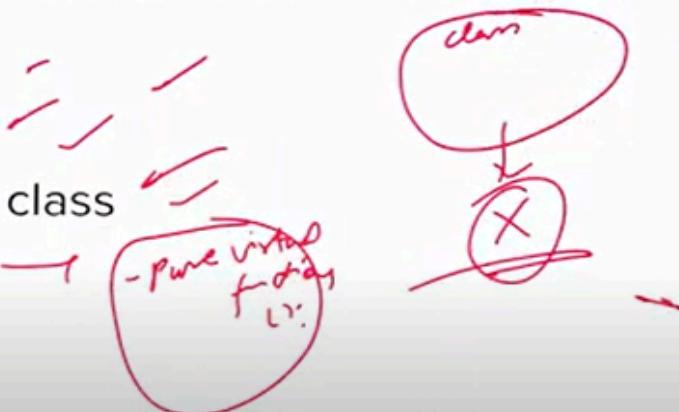
Objects are instances of the class.

Q3:

Problem 3

Which class cannot create its instance?

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



Q4:

Problem 4

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

cln & friend void fun()

friend() function is not a member of the class.

Static function is the member of the class only, not the object. yah hr object ke liye same hota hai.

Constant functions are nothing but the every parameter of the function are constant, we can not alter any attribute of the function.

Q5:

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?

Input

pre post
++ ++ (higher)

① Addition

② - ft shift)

③ -

Two:

One obj=Two(); ==>

Here we are not creating a pointer to an object.

Here we are creating an object of the base class and we are instantiating it with the object of the derived class.

This is **default copying**, it does not have any pointer and address.

Here overriding will not take place.

One `*obj=new Two();` ==> here we are creating the pointer of the base class and we are assigning the address of the derived class object.

Here overriding and everything come into picture.

Note:

- + Have higher priority than <<
 $2 + 4 + 3 << 2$
 $\Rightarrow (2 + 4 + 3) << 2$
 $\Rightarrow 9 << 2$
 $\Rightarrow 36$

Q6:

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?

$$10771 = 5 \\ 571 = ②$$

Here we are using the pointer of base class and that pointer are holding the address of the object of the child class.

So here method overriding will be used.

=====

50 Practice Problems in C++ OOPs - Part II | L:30

=====

Q1:

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

Q2:

Problem

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

char *p[] = {"HELLO,", "THIS", "IS", "JAY"};
char **q[] = {p+3, p+2, p+1, p};
char ***r = q;

int main()
{
    cout<<***++r<<" ";
    cout<<*--*++r+2<<" ";
    cout<<*r[-2]+1<<" ";
    cout<<r[-1][-1]+1<<endl;
    return 0;
}
```

Output ?

IS LLO, AY HIS

Q3:

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



Protected members are accessible to the child.

Q4:

Problem

Which of the following operators can be overloaded ?

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

All type of the assignment operator can be overloaded.

Q5:

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**

50 Practice Problems in C++ OOPs - Part III | L:31

Q1:

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 'B'

Default inherits private mode

Here class D inherits class B in private mode, so all the members of the class B which are inherited will become private.

constructor D() wants to call the C and D, it can call the constructor of class C because it is public, but it can not call D because it is private.

Yadi parent private inheritance kiya ho to , parent ka pointer child ka address nahi rakh sakta.

Therefore, B *b=new D();

Q2:

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

We can not create the object of the abstract class.

Q3:

Problem 3

```
class B{
public:
    virtual ~B(){
        cout<<"B Virtual Function"<<endl;
    }
};

class C{
public:
    virtual ~C(){
        cout<<"C Virtual Function"<<endl;
    }
};

class D: public B, public C{
public:
    ~D(){
        cout<<"Derived destructor"<<endl;
    }
};

int main(){
    B *b=new D();
    delete b;
    return 0;
}
```

C

Op:==> destructor can be virtual(see lec remaining of Cpp lec)

Derived destructor

C virtual function

B virtual function

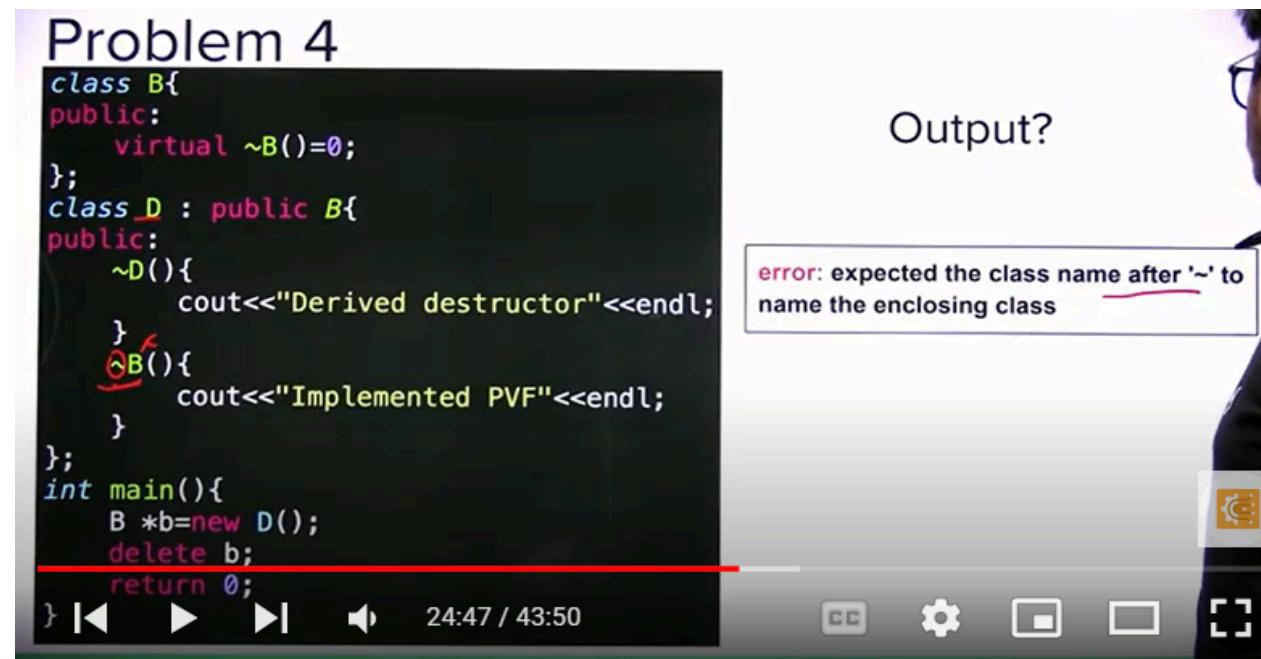
Q4:

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



Note:

You can not define the constructor and destructor of any other class into any other class.
Ie. A class can not hold any constructor and destructor of any other other class.

So class D can not hold the destructor of class B.

We can implement the ~B() outside of any class.

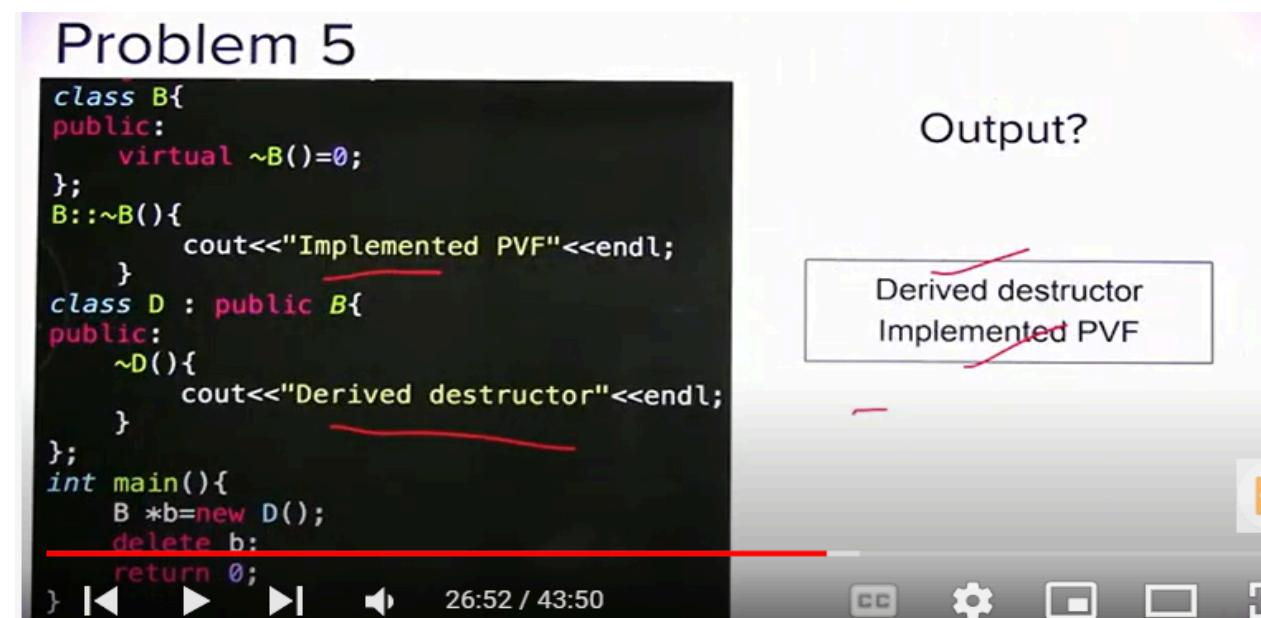
Q5:

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



Q6:

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

Annotations:
→ Compile time / static
→ Run time / dynamic
→ Method overriding
Virtual fns
↳ VTABLE - upto



Compile time polymorphism can be obtained by using
fo,00,And template .

Q7:

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**

Q8:

Problem 8

```
#include <bits/stdc++.h>
using namespace std;

class Foo{
public:
    Foo(int i=0){ _i=i;}
    void f(){
        std::cout<<"Hello!"<<std::endl;
    }
private:
    int _i;
};

int main() {
    Foo *p = 0;
    p->f();
    return 0;
}
```

Output: Hello!

Q9:

Problem 9

```
#include <bits/stdc++.h>
using namespace std;

class Foo{
public:
    int f(string str){
        cout<<str<<endl;
        return 1;
    }
};

int main() {
    int (Foo::*fptr)(string) = &Foo::f;
    Foo obj;
    (obj.*fptr)("Hello");
    Foo* p=&obj;
    (p->*fptr)("World");
    return 0;
}
```

Handwritten notes on the right side of the screen:

- >Hello
- World
- +fptr = f
- (obj.*fptr)("Hello")
- (p->*fptr)("World")

50 Practice Problems in C++ OOPs - Part IV | L:32

Q1:

Problem 1

Which of the following is not a type of constructor?

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

Q2:

Problem 2

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

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

Q3:

Problem 3

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

- A. **B obj = D();** ✓ correct
- B. D obj = B();
- C. **B *ptr; D obj;** ✓
 ptr = &obj;
- D. D *ptr; B obj;
 ptr = &obj;

Q4:

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

Activate

Q5:

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();
```

26 45



Here we are passing the obj2 value \Rightarrow it will be the pass by value

So changes into the "ob" will not affect the obj2, since another copy of the "ob" will be created.

If we use "&" value, then "ob" will point to the same object obj2, no separate object will be created.

Q6:

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

Q7: reverse the number without converting it into the string.

```
Int8_t,int16_t,int32_t,int64_t  
uint8_t,uint16_t,uint32_t,uint64_t
```

Problem 7

```
#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 = 16337893829345

Output?

n integer
 $n \leq 10^9$ $n \leq 10^9$

```
9 #include <iostream>  
10 using namespace std;  
11 uint64_t reverse(uint64_t n)  
12 {  
13     uint64_t result=0;  
14     while(n!=0)  
15     {  
16         result=result*10+n%10;  
17         n=n/10;  
18     }  
19     return result;  
20 }  
21 int main()  
22 {  
23     cout<<reverse(1234567899999);  
24  
25     return 0;  
26 }  
27
```

9999987654321

```
...Program finished with exit code 0  
Press ENTER to exit console.
```

Q8:

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?

A::a=4 ✓
B obj1(1, 4); b=4
cout<<obj1.fun(); 16
obj1.fun(1, 4); 4

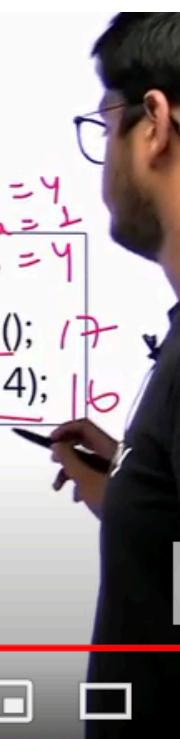
Q9:

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;
    }
};
```

B *ptr = &obj
B obj = B() Output?

A::a=4
B obj1(1, 4); b=4
cout<<obj1.fun(); 17
obj1.A::fun(1, 4, 4); 16



38:36 / 40:04

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

} |◀ ▶| ⏪ ⏩ 🔍 19:22 / 57:50



===== 50 Practice Problems in C++ OOPs - Part V | L:33 =====

Q1:

Problem 1

Which of the following features must be supported by any programming language to become a pure object-oriented programming language?

- A. Encapsulation
- B. Inheritance
- C. Polymorphism
- D. All of the above

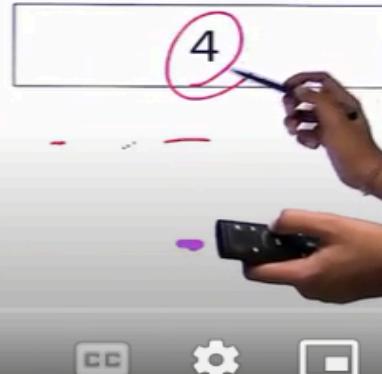
Q2:

Problem 2

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

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

Output?



`sizeof(class)==>`give the size of any object present into the class.

Static member of the class does not belong to any object of the class.

Q3:

Problem 3

```
#include<iostream>
using namespace std;
int main(){
    int a = 2.1;
    float b = 2.1;
    if(a==b) cout<<"Equal"<<endl;
    else      cout<<"Not Equal"<<endl;
    return 0;
}
```

Guess the output of this program

Output?

Not Equal

Q4:

Problem 4

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

30:18 / 57:50

Output?

error: no member named 'print'
in 'Host'

Class host does not have any right on the method print(), it belongs to nested class only.

Q5:

Problem 5

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

33:10 / 57:50

Output?

Hello world

Q6:

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?

class Host {
 int a; }
Host::Nested foo;
cout << foo.get();
This is valid

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

You can not create the object of the class before defining it completely.

Q7:

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;  
}
```

Q7 contd - L

Output?

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

Q8:

Problem 8

```
#include<iostream>
using namespace std;
int fun1(int);
int fun2(int);
int fun1(int n){
    if(n<=1) return 1;
    return fun2(n-2) + fun1(n-1);
}
int fun2(int n){
    if(n<=0) return 0;
    if(n%3==0) return fun2(n/2);
    else        return fun1(n-3);
}
int main(){
    int n;
    cin>>n;
    cout<<fun2(n)<<endl;
    return 0;
}
```

Calculate the output on n = 13

Output

14

