**Practice problems on Inheritance**

**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:** A B//Base class constructor will be called first.

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:** A 4 B 5

**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:** C A B

**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:** A5CB9

**Problem 5a:**

#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 be explicitly initialize the base class ‘A’ which doesn’t have a default constructor.

**Problem 5b:**

#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;

}

**Output:** A12B12

**Problem 6a:**

#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 6b:**

#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:** A9B5C2

**Problem 6c:**

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: A9B5 Hello World C2

Problem 7a:

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;

}

};

Int main(){

C obj(2,5,9);

Return 0;

}

**Output:** A9B5 Hello World C2

**Problem 7b:**

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;

}

};

Int main(){

A a(20);

a.get();

Return 0;

}

**Output:** A20 Hello world

**Problem 7c:**

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;

}

};

Int main(){

B obj2;

Obj2.get();

Return 0;

}

**Output:** Error: ‘get’ is a protected member of ‘A’.

**Problem 8:**

Class A{

Public: ~A(){

Cout<<”A”;

}

};

Class B{

Public: ~B(){

Cout<<”B”;

}

};

Class C:public A,B{

Public: ~C(){

Cout<<”C”;

}

};

Int main(){

C c;

Return 0;

}

**Output:** CBA

**Problem 9:**

Class base{

Public: 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 10a:**Which specifier makes all the data members and functions of base class inaccessible by the derived class

1. Private
2. Protected
3. Public
4. Both private and protected

**ANS: A) Private**

**Problem 10b:**

Class A{

Public: int a;

Void change(int i){

a=i;

}

Void value\_of\_a(){

Cout<<a;

}

};

Class B: public A{

In a=15;

Public: void print(){

Cout<<a;

}

};

Int main(){

B b;

b.change(10);

b.print();

b.value\_of\_a();

return 0;

}

**Output:** 15 10