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

class Base{
private:
     int t;
public:
     Base() {
          cout << "Base constructor-0 param" << endl;
          t=0;
     }
     Base(int u) {
          cout << "Base constructor-1 param" << endl;
          t=u;
     }
     Base(const Base& b){
          t=b.t;
          cout << "Base copy constructor" << endl;
     }
     ~Base(){
          cout<<"Base destructor" << endl;
     }
};
```

```
class DerivedA: public Base {
public:
     DerivedA(){
          cout<<"DerivedA constructor-0 param"<<endl;
          a=0;
     }
     DerivedA(int x) {
          cout<<"DerivedA constructor-1 param"<<endl;
          a = x;
     }
     DerivedA(int u, int x) {
          cout<<"DerivedA constructor-2 params"<<endl;
          a = x;
     }
     DerivedA(const DerivedA& da){
          a=da.a;
     }
     ~DerivedA(){
          cout<<"DerivedA destructor" << endl;
     }
     void print() {
          cout << "DerivedA = " << a << endl;
     }
private:
     int a;
};
```

```
class DerivedB: public Base {
public:
	DerivedB() {
		cout << "DerivedB constructor-0 param" << endl;
		b = 0;
	}
	DerivedB(float x) {
		cout << "DerivedB constructor-1 param" << endl;
		b = x;
	}
	DerivedB(int u, float x):Base(u) {
		cout << "DerivedB constructor-2 params" << endl;
		b = x;
	}
	DerivedB(const DerivedB& db):Base(db) {
		cout << "DerivedB copy constructor" << endl;
		b=db.b;
	}
	~DerivedB(){
		cout<<"DerivedB destructor" << endl;
	}
	void print() {
		cout << "DerivedB = " << b << endl;
	}
private:
	float b;
};
```

```
void f(Base  b){
	cout<<"Function f" << endl;
}
```

```
int main(){
	Base b1;
	Base b2(3);
	Base*pb=new Base(1);

	DerivedA da1;
	DerivedA da2(2);
	DerivedA da3(2,3);
	DerivedA da4(da2);
	pb=new DerivedA(4,5);

	DerivedB db1;
	DerivedB db2(2.7);
	DerivedB db3(3,2.7);
	DerivedB db4(db2);
	pb=new DerivedB(4,5);

	f(b1);
	f(da1);
	f(db1);
	delete pb;
	return 0;
}
```