1.- ¿Qué muestra por pantalla?

```
#include <iostream>

class A{
public :
   int Valor=4;
   A(){}

};

class B: public A{
public:
   int Valor=2;
   B(int x):A(){Valor=x;}
};

int main()
{

   B objB(3);

   std::cout<<objB.Valor;
   std::cout<<objB.A::Valor;

   return 0;
}
```

34
43
Error de compilación
33

2.-¿Qué muestra por pantalla?

```
#include <iostream>

class A{
public :
   int Valor;

   A(int x){Valor=x;}
   int X(){return Valor*10;}
};

class B: public A{
public:
   B(int x):A(x){}
   int X(){return Valor*1;}
};

int main()
{

   B objB(2);
   std::cout<<objB.X()<<std::endl;

   return 0;
}
```

2
20
Error de compilación
Un valor indeterminado

3.- ¿Qué muestra por pantalla?

```
#include <iostream>

class A{
public:
   int Valor=4;
};

class B: public A{
public:
   B(int x){Valor=x;}
};

int main()
{
   B objB(3);
   std::cout<<objB.Valor<<std::endl;
   return 0;
}
```

3
Error de compilación
4
44

4.- ¿Qué muestra por pantalla?

```
#include <iostream>

class A{
public:
   int Valor=4;
};

class B: private A{
public:
   B(int x){Valor=x;}
};

int main()
{
   B objB(3);
   std::cout<<objB.Valor<<std::endl;
   return 0;
}
```

3
Error de compilación
4
44

Comentado [4]: En este caso heredamos de manera privada una variable publica=> Valor en B estará dentro del "saco" privado

El constructor no da error, ya que puede acceder a su parte privada.

El error viene en el main al intentar acceder a la parte privada desde el exterior de la clase

5.- ¿Qué muestra por pantalla?

```
#include <iostream>

class A{
private:
   int Valor=4;
};

class B: public A{
public:
   B(int x){Valor=x;}
};

int main()
{
   B objB(3);
   std::cout<<objB.Valor<<std::endl;
   return 0;
}
```

3
Error de compilación
4
44

Comentado [5]: En este caso la clase hereda públicamente una variable privada => Valor no es accesible en la clase derivada

Además en el main estamos intentando acceder a una variable derivada.

Doble error

6.-¿Qué muestra por pantalla?

```
#include <iostream>

class A{
public :
   A(){std::cout<<"HolaA";}
   ~A(){std::cout<<"AdiosA";}
};

class B: public A{
public:
   B(){std::cout<<"HolaB";}
   ~B(){std::cout<<"AdiosB";}
};

int main()
{
   B objB;

   return 0;
}
```

HolaBHolaAAdiosBAdiosA
HolaAHolaBAdiosAAdiosB
HolaAHolaBAdiosBAdiosA
Error de compilación

Comentado [6]: Primero constructor de la base, luego el de la derivada.

Primero destructor de la derivada, luego de la base

7.- ¿Qué muestra por pantalla?

```
#include <iostream>

class A{
public :
   A(int x){std::cout<<"HolaA"<<x;}
   ~A(){std::cout<<"AdiosA";}
};

class B: public A{
public:
   B(int y){std::cout<<"HolaB"<<y;}
   ~B(){std::cout<<"AdiosB";}
};

int main()
{
   B objB(2);

   return 0;
}
```

Error de compilación
HolaAHolaBAdiosBAdiosA
HolaAHolaBAdiosAAdiosB
HolaBHolaAAdiosBAdiosA

8.- ¿Qué muestra por pantalla?

```
#include <iostream>

class A{
public :
   A(int x){std::cout<<"HolaA"<<x;}
   ~A(){std::cout<<"AdiosA";}
};

class B: public A{
public:
   B(int y):A(y){std::cout<<"HolaB"<<y;}
   ~B(){std::cout<<"AdiosB";}
};

int main()
{
   B objB(3);

   return 0;
}
```

Error de compilación
HolaA3HolaB3AdiosBAdiosA
HolaB3HolaA3AdiosBAdiosA
HolaA3HolaB3AdiosAAdiosB

Comentado [8]: En este caso el constructor de la derivada si llama al constructo de la base=> contructor base; constructor derivada; destructor derivada; destructor base

9.- ¿Qué muestra por pantalla?

```
#include <iostream>

class A{
public :
   A(){std::cout<<"HolaA";}
   ~A(){std::cout<<"AdiosA";}
};

class B: public A{
public:
   B(int y){std::cout<<"HolaB"<<y;}
   ~B(){std::cout<<"AdiosB";}
};

int main()
{
   B objB(2);

   return 0;
}
```

Error de compilación
HolaAHolaBAdiosBAdiosA
HolaBHolaAAdiosBAdiosA
HolaAHolaBAdiosAAdiosB

Comentado [9]: El constructor con argumento de la derivada "sabe" llamar al constructor por defecto de la base sin indicarlo explicitamente

=> constructor de la base; constructor de la derivada; destructor de la derivada; destructor de la base

10.- ¿Qué muestra por pantalla?

```
#include <iostream>

class Base
{
   public:
      Base() {std::cout << "BaseON_";}
      ~Base() {std::cout << "BaseOFF_";}
};

class A : public Base
{
   public:
      A(){std::cout << "AON_";}
      ~A() {std::cout << "AOFF_";}
};

class B : public A
{
public:
   B() {std::cout << "BON_";}
   ~B() {std::cout << "BOFF_";}
};

int main()
{
   B d;

   return 0;
}
```

BaseON_AON_BON_BOFF_AOFF_BaseOFF
BON_AON_BaseON_BaseOff_AOFF_BOFF
Error de compilación
BaseON_BON_AON_BOFF_AOFF_BaseOFF

11.- ¿Qué muestra por pantalla?

```
#include <iostream>
#include <memory>

struct B
{
   int x{3};
   B(int v){x=v;}
};


int main()
{
   B d(4);

   std::shared_ptr<B> punt=std::make_shared<B>(d);
   std::cout<<punt->x;
   d.x=8;
   std::cout<<punt->x;

   return 0;
}
```

```
44
33
48
38
```

12.-¿Qué muestra por pantalla?

```
#include <iostream>
#include <memory>

struct B
{
   int x{3};
   B(int v){x=v;}
   ~B(){};
};


int main()
{
   B d(4);

   std::shared_ptr<B> punt(&d);
   std::cout<<punt->x;
   d.x=8;
   std::cout<<punt->x;

   return 0;
}
```

```
44
33
48
38
```

Comentado [12]: En esta ocasión inicializamos el puntero con la dirección del objeto => Apuntamos a su dirección los cambios que hagamos en d se "verán" en punt y al contrario

=> 48