1.- ¿Qué muestra por pantalla?

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

using namespace std;

int main()
{
    int k{3};

    auto p=make_shared<int>(k);
    auto p1=make_shared<int>(k);

    cout<<*p1<<*p;

    *p+=1;
    cout<<*p1<<*p;
    *p1+=1;
    cout<<*p1<<*p;

    return 0;
}
```

333444
334344
333445
334455

2.- ¿Qué muestra por pantalla?

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

using namespace std;

int main()
{
    int k{3};

    auto p=make_shared<int>(k);
    auto p1=p;

    cout<<*p1<<*p;

    *p+=1;
    cout<<*p1<<*p;
    *p1+=1;
    cout<<*p1<<*p;

    return 0;
}
```

334455
334344
333445
333444

3.- ¿Qué opción de las siguientes es verdadera?

```cpp
#include <iostream>

class Foo
{
   int var{3};
   std::string who(){return "Foo";}
};

class Fii{
   int var{2};
   Foo Obj;
};

int main()
{
   Fii a;
   std::cout<<a.var<<a.Obj.who();

   return 0;
}
```

Error de compilación porque no puedo acceder al método de un objeto encapsulado en otra clase [a.Obj.who()]
Error de compilación al instanciar el objeto a, porque Fii no tiene constructor público.
Error de compilación porque Foo no puede encapsular un objeto de otra clase
Error de compilación porque Fii tiene sus miembros privados
El programa muestra por terminal 2Foo

Comentado [3]: Foo y Fii no tienen ningún problema de compilación.
Son 2 clases con todos sus miembros privados por defecto.

No hay problema en que una clase encapsule un objeto (un objeto es una variable de un tipo clase determinado, y la clase encapsula variables y funciones)

Para acceder a un método de una clase encapsulado en otra usamos el operador . como siempre.

Otra cosa es que no pueda acceder a ellos porque desde el "exterior" porque sean privados

4.- ¿Qué muestra por pantalla?

```
#include <iostream>

struct Foo
{
    int var;
    Foo* p;
    Foo(){var=0;p=nullptr;};
    Foo(int a, Foo* s){var=a;p=s;};
};

int main()
{
    Foo a,b(a.var+1,&a),c(b.var+1,&b);
    Foo* aux{&c};

    while(aux->p!=nullptr)
    {
        std::cout<<aux->var;
        aux=aux->p;
    }

    return 0;
}
```

21
12
012
210

5.- ¿Qué muestra por pantalla?

```
#include <iostream>

int foo(float b)
{
    if(b>3)
    {
        return 3*b;
    }
    else
    {
        throw std::string{"error"};
    }
};

int main()
{
    try{
        std::cout<<foo(3.1);
        std::cout<<foo(3.0);
        std::cout<<foo(6);
    }
    catch(std::string e)
    {
        std::cout<<e;
    }
    std::cout<<"fin";
}
```

9errorfin
9.3errorfin
9error18fin
9.3error18fin

6.- ¿Qué muestra por pantalla?

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

class Foo
{
    public:
    Foo(int a){std::cout<<a;};
    ~Foo(){std::cout<<"Foo";};
};

int main()
{
    std::cout<<1;
    std::shared_ptr<Foo> p;
    std::cout<<2;
    p=std::make_shared<Foo>(5);
    std::cout<<3;
}
```

1253Foo
15253Foo
15253
1253

7.- ¿Qué muestra por pantalla?

```
#include <iostream>

struct Foo
{
    int x,y;
};

Foo suma(Foo a, Foo b)
{
    a.x=a.x+b.x;
    a.y=a.y+b.y;

    return a;
}

int main()
{
    Foo t1{1,1};
    Foo t2{2,2};

    std::cout<< suma(t1,t2);
}
```

33
3
Error cout no sabe imprimir una estructura Foo
1122

8.- ¿Qué muestra por pantalla?

```cpp
#include <iostream>
#include <memory>
#include <vector>
using namespace std;
int main()
{
	auto a=make_shared<std::vector<int>>(vector<int>{2,3,4});
	auto b=make_shared<std::vector<int>>(vector<int>{1,2,3});
	auto c=a;

	a=b;

	b->at(1)=0;

	std::cout<<a->at(1)<<b->at(1)<<c->at(1);

	return 0;
}
```

303
202
102
003

Comentado [7]: declaramos un puntero shared a un vector de int 2 3 4
declaramos otro puntero shared a un vector de int 1 2 3

declaramos un puntero c que es igual a

hacemos que a apunte a b

ponemos a 0 la posición 1 apuntada por b (a apunta al mismo sitio que b)

Mostramos posición 1 de a b y c

a y b en la posición 1 tiene un 0 => 00

c apunta donde apuntaba a al principio => 3

9.- ¿Qué muestra por pantalla?

```cpp
#include <iostream>
#include <memory>
#include <array>
using namespace std;
int main()
{
	auto a=array<shared_ptr<int>,3>{};
	auto b=make_shared<int>(1);
	auto c=make_shared<int>(2);
	auto d=b;

	a={b,c,d};
	cout<<*a.at(0)<<*a.at(1)<<*a.at(2);
	*b=5;
	cout<<*a.at(0)<<*a.at(1)<<*a.at(2);
	return 0;
}
```

123123
121121
121525
121521

Comentado [8]: Tenemos un array de 3 punteros inteligentes a int
Tenemos un puntero b shared que apunta a 1
Tenemos un puntero c shared que apunta a 2
y otro puntero d que apunta a donde apunte b

Inicializamos a con los punteros b c d

Mostramos 121

Modificamos el contenido al que apunta b por un 5 (b y d apuntan al mismo sitio)

Mostramos 525

10.-¿Qué muestra por pantalla?

```
#include<iostream>

class MiClase
{
private:
   int valor;
public:
   MiClase(){std::cout<<"1";}
   ~MiClase(){std::cout<<"2";}
};

void fun(void)
{
   MiClase Cfun;
   std::cout<<"3";
}

int main()
{
   MiClase C1;
   std::cout<<"4";
   fun();
   std::cout<<"5";
   return 0;
}
```

1413252
1413522
Error de compilación
1241235