1.- ¿Qué muestra por pantalla?

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

int main(){
   auto a = std::make_shared<int>(10);
   auto b = a;
   *b = 5;
   *a = 3;
   std::cout << *a << "_" << *b;
   return 0;
}
```

3_3
3_5
10_10
5_5

2.- ¿Qué muestra por pantalla?

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

int main(){
   auto a = std::make_shared<int>(10);
   auto b = std::make_shared<int>(*a);
   *b = 5;
   *a = 3;
   std::cout << *a << "_" << *b;
   return 0;
}
```

3_3
3_5
10_10
5_5

Comentado [2]: a apunta a una posición de memoria que contiene un 10

b apunta a otra posición de memoria que contiene otro 10. (Pero no son la misma posición de memoria. En el caso anterior si apuntaban a la misma dirección de memoria porque b=a, aquí no)

Asignamos en la indirección de b un 5
Asignamos en la indirección de a un 3

Mostramos 3_5

3.- ¿Qué muestra por pantalla?

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

int main(){
    auto a = std::make_shared<int>(10);
    auto b = std::make_shared<int>(10);
    if(a == b)
    {
        std::cout << 1;
    }
    else
    {
        std::cout << 2;
    }

    return 0;
}
```

1
2
12
Error de compilación

Comentado [3]: a apunta a una posición de memoria que contiene un 10

b apunta a otra posición de memoria que contiene otro 10

Pero a y b son diferentes... => Mostramos 2

4.- ¿Qué muestra por pantalla?

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

auto flip(const std::shared_ptr<int*>& v)
{
    std::cout<<**v;
    **v+=1;;
}

int main(){
    int valor{55};
    auto a = std::make_shared<int*>(&valor);
    flip(a);
    std::cout<<**a;
    return 0;
}
```

5556
5555
Valores hexadecimales
Error de compilación

5.- ¿Qué muestra por pantalla?

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

using namespace std;

struct Cosa{
   int entero1{11};
   int entero2{22};
   Cosa(){entero1++;entero2++;};
};

auto foo(const std::unique_ptr<Cosa>& c)
{
   c->entero2++;
   c->entero1++;
}

int main(){
   Cosa Elemento;
   auto pElemento = std::make_unique<Cosa>(Elemento);

   std::cout<<pElemento->entero1<<pElemento->entero2;

   foo(pElemento);

   std::cout<<pElemento->entero1<<pElemento->entero2;

   return 0;
}
```

12231324
Error de compilación
11221223
12231223

6.-

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

using namespace std;

struct Cosa{
   int entero1{11};
   int entero2{22};
   Cosa(){entero1++;entero2++;};
};

auto foo(const std::unique_ptr<Cosa>& c)
{
   c->entero2++;
   c->entero1++;
}

int main(){
   Cosa Elemento;
   auto pElemento = std::make_unique<Cosa>(Elemento);

   foo(pElemento);

   std::cout<<pElemento.entero1<<pElemento.entero2;
   std::cout<<Elemento.entero1<<pElemento.entero2;

   return 0;
}
```

12231324
Error de compilación
11221223
12231223

Comentado [6]: No podemos acceder a los miembros de un objeto a través de un puntero con . necesitamos ->

7.-

```
#include <iostream>

class flip{
	public:
	flip(){std::cout<<"1";};
	flip(const flip& f){std::cout<<"2";};
	~flip(){std::cout<<"3";};
};

int main()
{
	flip f1;
	if(true)
	{
	    flip f2{f1};
	}

	return 0;
}
```

1233
Nada
Error de compilación
1133

8.-

```
#include <iostream>

struct flip{
	flip(){std::cout<<"2";};
	~flip(){std::cout<<"1";};
};

int main()
{
	flip f1;

	return 0;
}
```

21
12
Nada
Error de compilación

9.-

```
class flip{
      flip(){std::cout<<"2";};
      ~flip(){std::cout<<"1";};
};

int main()
{
      flip f1;

      return 0;
}
```

21
12
Nada
Error de compilación

Comentado [9]: Una clase por defecto tiene todos sus miembros por defecto privados.

Al crear un objeto se llama al constructor, como es privado no podemos acceder desde fuera y da error de compilación.

10.-

```cpp
#include <iostream>
#include <memory>
#include <vector>

struct flip{
      int v{3};
      flip(){v=1;}
};

int main()
{
      auto p=std::make_shared<flip>();

      std::vector<std::shared_ptr<flip>> v;

      for(int i{1};i<5;i++)
      {
          if(i%2==true)
          {
               p->v;
          }
          else
          {
               p->v+=1;
          }
          v.push_back(p);
      }

      for(auto e:v)
      {
          std::cout<<e->v;
      }

      return 0;
}
```

Error de compilación
Un montón de valores en hexadecimal
3333
1223

Comentado [10]: Tenemos una estructura con v inicializada a 3. Y un constructor que la escribe a 1

En el main creamos un puntero a flip que usa el constructor por defecto => p->v valdrá 1.

Creamos un vector de punteros a flip

Bucle desde i=1 mientras i<5 i++
Si el índice es impar accedemos al campo v apuntado por el puntero (y no hacemos nada con el...)
Si el índice es par=> sumamos 1 a la v

Mostramos el valor v y añadimos el puntero p al vector v

Vuelta i=1=> guardamos p a flip con v=1
Vuelta i=2=>guardamos p a flip con v=2
Vuelta i=3=>guardamos p a flip con v=2
Vuelta i=4=>guardamos p a flip con v=3

Antes de terminar recorremos el vector de punteros y vamos accediendo a v que vale 3 en todas las ocasiones... ya que todos los punteros añadidos al vector apuntan al mismo objeto.

```
////////////////////////////////////////////////////////////////
Jugar con este concepto para próximo test

#include <iostream>
#include <memory>

using namespace std;

int main()
{
    int k{3};

    auto p=make_shared<int>(k);
    auto p1=make_shared<int>(k);
    auto p2=p1;

    //auto p2=make_shared<int*>(&k);

    cout<<&p<<"_"<<p.get()<<endl;
    cout<<&p1<<"_"<<p1.get()<<endl;
    cout<<&p2<<"_"<<p2.get()<<endl;

    *p+=1;
    cout<<*p<<"_"<<p.get()<<endl;
    cout<<*p1<<"_"<<p1.get()<<endl;
    cout<<*p2<<"_"<<p2.get()<<endl;
    *p1+=1;
    cout<<*p<<"_"<<p.get()<<endl;
    cout<<*p1<<"_"<<p1.get()<<endl;
    cout<<*p2<<"_"<<p2.get()<<endl;
    *p2+=1;
    cout<<*p<<"_"<<p.get()<<endl;
    cout<<*p1<<"_"<<p1.get()<<endl;
    cout<<*p2<<"_"<<p2.get()<<endl;

    return 0;
```

```
}
```