3. Indicar la salida por pantalla (2 puntos-20 minutos)

```
#include <iostream.h>
template <class S> void Funcion_B(S& a, S& b);
template <class T> void Funcion_A (T *vector, int num);
void main(void)
{
	float mivector[10]={2,12,3,8,1,3,15,6,4,1};
	Funcion_A(mivector,5);
	Funcion_A(mivector+5,5);

	for(int i=0;i<10;i++)cout<<mivector[i]<<endl;
}
template <class S> void Funcion_B(S &a, S &b)
{
	S temp;
	temp = a;
	a = b;
	b = temp;
}
template <class T> void Funcion_A (T *vector, int num)
{
	int i,j;
	for(i=0;i<num-1;i++)
	for(j=i+1;j<num;j++)
		if(vector[i]>vector[j])
			Funcion_B(vector[i],vector[j]);
}
```

Impresión por pantalla

4. Ejercicio de programación (2,5 puntos-50 minutos)

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

void copiaCadena(char *destino, char *origen, int max){
int i=0,n;
n=strlen(origen);
if(max<1)return;
if(n>max-1){
       n=max-2;
       destino[max-1]=0;
       }
for(i=0;i<n+1;i++)destino[i]=origen[i];
}

class ObraDeArte
{
 virtual void imprimeTipo()=0;
 virtual void imprimeDatosAdicionales(){}
 char titulo[100];
 char autor[100];
public:
 ObraDeArte(char *_titulo, char *_autor);
 void imprime();
};

void ObraDeArte::imprime(){
 cout<<"____________________"<<endl;
 imprimeTipo();
 cout<<"Titulo: "<<titulo<<endl;
 cout<<"Autor: "<<autor<<endl;
 imprimeDatosAdicionales();
 }

//////////////////////////////////////
//////CODIGO A DESARROLLAR POR EL ALUMNO
//////////////////////////////////////

void main()
{
ObraDeArte *lista[4];
lista[0]=new Cuadro("Las Meninas","Velazquez");
lista[1]=new Libro("El Quijote","Cervantes",978);
lista[2]=new Estatua("La Piedad","Miguel Angel","Italia");
lista[3]=new Cuadro("Arlequin","Picasso");
for(int i=0;i<4;i++){
       lista[i]->imprime();
       delete lista[i];
       }
}
```

```
____________________
PINTURA
Titulo: Las Meninas
Autor: Velázquez
____________________
LIBRO
Titulo: El Quijote
Autor: Cervantes
número de Páginas: 978
____________________
ESTATUA
Titulo: La Piedad
Autor: Miguel Ángel
Pais: Italia
____________________
PINTURA
Titulo: Arlequín
Autor: Picasso
```

Dado el estracto de código mostrado, se pide:

1.-Definir el constructor de la clase ObraDeArte.
2.-Declarar y definir la clase Cuadro.
3.-Declarar y definir la clase Libro.
4.-Declarar y definir la clase Estatua.

De tal forma que la impresión obtenida por el main sea la que aparece en el recuadro.

La puntuación del ejercicio irá no sólo en funcion del resultado, sino de la correcta programación usando las herramientas propias de la POO.

5. Problema de Análisis y Diseño Orientado a Objetos (2.5 puntos - 50 minutos)

Se desea hacer una aplicación que sirva para calcular las nominas de una compañía. Al salario base de cada empleado hay que quitarle un 20% de retención del IRPF para calcular su salario neto.
Como existen diferentes políticas salariales en la empresa, se desea hacer un programa fácilmente extensible a nuevas políticas. De momento se pretende abordar dos de ellas:

- El sueldo ordinario
- El sueldo con bonus, consistente en aumentar el salario base (antes de la retención) un 35%.

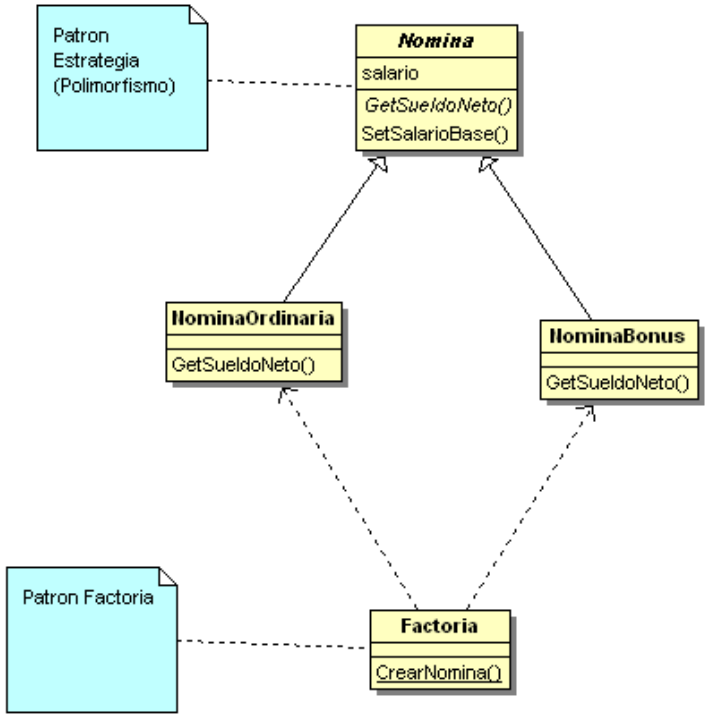
De momento se ha desarrollado el siguiente programa:

```
main()
{
	Nomina* nomina;
	cout<<"1. Nomina ordinaria"<<endl;
	cout<<"2. Nomina con bonus"<<endl;
	cin>>opcion;

	//Completar codigo aqui

	cout<<"Salario base: ";
	float salario;
	cin>>salario;
	nomina->SetSalarioBase(salario);
	float total=nomina->GetSueldoNeto();
	cout<<"El salario neto es: "<<total<<endl;
}
```

Se pide:

1. Diagrama de Clases de Diseño de una arquitectura que permita una fácil extensión a nuevas políticas. Indicar los patrones utilizados. (5 puntos)
2. Implementación en C++ de la solución, completando el main(). (5 puntos)

SOLUCION:




```
class Nomina
{
public:
	virtual float GetSueldoNeto()=0;
	void SetSalarioBase(float s)
	{
		salario=s;
	}

protected:
	float salario;

};
```

```
#include "Nomina.h"

class NominaBonus : public Nomina
{
public:
	float GetSueldoNeto()
	{
		return (salario*1.35f)*0.80f;
	}
};
```

```
#include "Nomina.h"

class NominaOrdinaria : public Nomina
{
public:
	float GetSueldoNeto()
	{
	return salario*0.80f;
	}
};
```

```
#include "Nomina.h"
```

```
class Factoria
{
public:
	static Nomina* CrearNomina(int tipo)
	{
	if(tipo==1)
		return new NominaOrdinaria();
	if(tipo==2)
		return new NominaBonus();

	return 0;
	}

};
```

```
#include "Nomina.h"
#include "Factoria.h"

int main()
{
	Nomina* nomina;
	cout<<"1. Nomina ordinaria"<<endl;
	cout<<"2. Nomina con bonus"<<endl;
	int opcion;
	cin>>opcion;

	//Completar codigo aqui
	nomina=Factoria::CrearNomina(opcion);
	if(nomina==0)
	{
		cout<<"Error tipo nomina"<<endl;
		return -1;
	}

	cout<<"Salario base: ";
	float salario;
	cin>>salario;
	nomina->SetSalarioBase(salario);
	float total=nomina->GetSueldoNeto();
	cout<<"El salario neto es: "<<total<<endl;

	return 0;
}
```