FUNDAMENTOS DE PROGRAMACIÓN (18/19)
Examen Teórico (Mod. A y B)
Puntuación máxima: 10 puntos
Duración: 2.5 horas

Puntuación:	Nombre del alumno:

1. (**2 puntos**) Realiza el seguimiento del código mostrado a continuación e indica qué muestran por pantalla los **cout** sombreados. Para ello se debe utilizar los datos de entrada indicados. Se suponen escritos todos los **#include** necesarios.

```
class Vector {
   float x, y;
public:
   Vector();
   void NuevosXY(float cx, float cy);
   void ComponentesXY (float &cx, float &cy);
   void Sumar(float cx, float cy);
   Vector Sumar (Vector v);
   int Cuadrante();
};

Vector::Vector() {
  x=y=0;
}

void Vector::NuevosXY(float cx, float cy) {
   x=cx;
   y=cy;
}

void Vector::ComponentesXY(float &cx, float &cy){
   cx=x;
   cy=y;
}
```

```
void Vector::Sumar(float cx, float cy){
   x+=cx;
   y+=cy;
}

Vector Vector::Sumar (Vector v){
   v.Sumar(x,y);
   return v;
}

int Vector::Cuadrante() {
   int c;
   if (x>0 && y>0)
       c=1;
   else if (x<0 && y>0)
       c=2;
   else if (x<0 && y<0)
       c=3;
   else
       c=4;
   return c;
}
```

```
int main() {
    Vector Conjunto[3], VSuma;
    float x, y;
    int i;

    for (i=0; i<3; i++) {
        cout << "Indique coordenadas x e y del vector: ";
        cin >> x >> y;
        Conjunto[i].NuevosXY(x, y);
    }

    i--;
    while (i>=0) {
       VSuma = VSuma.Sumar(Conjunto[i]);
       i--;
    }
    VSuma.ComponentesXY(x, y);
    cout << "El vector resultado es (" << x << ", " << y << ")" << endl;
    cout << "Dicho vector está en el cuadrante: " << VSuma.Cuadrante() << endl;

    return 0;
}
```

Datos de entrada – MODELO A:
3 1 1 2 -5 -2

Datos de entrada – MODELO B:
-2 1 2 3 4 -3

Solución:

Modelo A: El vector resultado es (-1, 1)
Dicho vector está en el cuadrante: 2

Modelo B: El vector resultado es (4, 1)
Dicho vector está en el cuadrante: 1

2. (**2 puntos**) Partiendo de las siguientes definiciones, completar el código de la función `main()` según se indica en la misma.

```
struct futbolista
{
   char nombre[51]; // Nombre del futbolista.
   int demarcacion; // 1 portero, 2 defensa, 3 central, 4 delantero
   float nomina;    // sueldo del futbolista.
};

struct equipo
{
   char nombre[51];          // Nombre del equipo de futbol.
   float presupuesto;        // Presupuesto del equipo.
   int numjugadores;         // Nº de jugadores en la plantilla del equipo.
   futbolista jugadores[25]; // Datos de cada jugador de la plantilla.
};

struct liga
{
   char temporada[14]; // Temporada de la liga. Ejemplo: "Liga 13-14"
   int numequipos;     // Nº de equipos participantes en la liga de esa temporada.
   equipo equipos[20]; // Clasificación final de los equipos de dicha liga, la primera posición
};                     // corresponde al equipo que ha quedado primero, y la última posición al
                       // que ha quedado último.
int main()
{  liga ligas[10]; // Información de cada liga jugada
   int Nligas;    // Número de ligas jugadas

/* Suponiendo que el vector ligas está completamente lleno de datos, implementar el código
necesario para dar respuesta a una de las siguientes peticiones:
 1.- Para cada liga, mostrar el nombre del equipo clasificado en el último lugar, así como el
 nombre de sus porteros.

 2.- Mostrar la media de los sueldos de los jugadores cuyo equipo ha ganado en cada liga.
*/
    return 0;
}
```

PETICIÓN 1:

```
    for (int l=0; l<Nligas; l++)
    {
        int Pos=ligas[l].numequipos-1;
        equipo e=ligas[l].equipos[Pos];
        cout<< e.nombre;
        for (int j=0; j<e.numjugadores; j++)
            if (e.jugadores[j].demarcacion==1)
                cout<<e.jugadores[j].nombre;
    }
```

PETICIÓN 2:

```
    for (int l=0; l<Nligas; l++)
    {
        equipo e=ligas[l].equipos[0];
        float suma=0;
        for (int j=0; j<e.numjugadores; j++)
            suma+=e.jugadores[j].nomina;
        suma/=e.numjugadores;
        cout<<"Media de las nominas equipo "<<l+1<<" "<<suma<<endl;
    }
```

3. Las siguientes estructuras de datos son utilizadas para almacenar la información de terminales móviles cuando sus dueños viajan por las carreteras de España. A lo largo de las carreteras se sitúan, un conjunto de torres de comunicación que registran los datos de los móviles que se encuentran en su radio de acción, que es de 50Km.

   Para almacenar la información de un terminal móvil, se utiliza la estructura de tipo registro **Terminal**. La clase **Torre** almacena los datos de todos los móviles que están bajo su cobertura. Una torre necesita, además, la información del nombre de la **carretera** y el **kilómetro** donde está situada. **Los métodos de esta clase se consideran implementados**.

```
typedef char Cadena [50];

struct Terminal {
    int Numero;     // Numero del Móvil
    Cadena Modelo; // Modelo del Móvil
    float Bateria; // Nivel de la batería
};
class Torre  {                  //Torre de Comunicaciones
    Cadena Carretera;           //Nombre de la carretera
    int Kilometro;              //Kilómetro dentro de la carretera
    Terminal Moviles[10000];    //Terminales móviles registrados en la torre
    int NMoviles;               //Número de Terminales registrados
public:
    void getCarretera(Cadena Nombre);  //Devuelve el atributo carretera
    int getKilometro();          //Devuelve el atributo Kilometro
    int getNoMoviles();          //Devuelve el atributo NMoviles
    Terminal getMovil(int Pos);  //Devuelve los datos del terminal que está en la posición
                                 //del vector que indica Pos
    int BuscarMovil(int Numero); //Devuelve la posición dentro del vector Moviles donde está
                                 //situado el móvil cuyo número coincide con el pasado por
                                 //parámetro
};
```

(3 Puntos) Implementar la siguiente función genérica según se indica en el comentario:

```
/* Dada una red de torres de comunicación (parámetros Red y NTorres), la función devolverá la
posición dentro del vector Red, de la torre que está situada en el kilómetro y carretera que se
indican por parámetro. Si no existe ninguna torre en la carretera y kilómetro indicados, la
función devolverá -1 */

int BuscarTorre(Torre Red[1000], int NTorres, Cadena Carretera, int Kilometro)
{

}
```

(3 Puntos) Implementar la función main según se indica en el comentario:

```
int main()
{
    Torre RedEspana[1000]; //Conjunto de torres de comunicación de toda España
    int NTorres;           //Número de Torres


/* Dado un número de móvil introducido por teclado, mostrar el nivel de batería registrado por
las torres de comunicación de la autopista AP-7, cuya longitud es de 1109 Kilómetros. En esta
autopista hay una torre de comunicación cada 50 kilómetros.

En caso de que ninguna torre haya registrado un móvil con el número introducido, se debe indicar
mediante un aviso por pantalla */

    return 0;
}
```

```
int BuscarTorre(Torre Red[1000], int NTorres, Cadena Carretera, int Kilometro)
{
    Cadena Nombre;
    int Km;
    int Pos=0;
    bool Encontrado=false;
    while (Pos<NTorres && !Encontrado)
    {
        Km=Red[Pos].getKilometro();
        Red[Pos].getCarretera(Nombre);
        if (strcmp(Carretera, Nombre)==0 && Km==Kilometro)
            Encontrado=true;
        else
            Pos++;
    }
    if (!Encontrado)
        Pos=-1;
    return Pos;
}



int main()
{

    Torre RedEspana[1000];
    int NTorres;


    bool Registrado=false;
    int NumeMovil;
    cout << "Introduce el número de móvil a buscar ";
    cin>>NumeMovil;
    for (int km=0; km<1109; km+=50)
    {
        int T=BuscarTorre(RedEspana,NTorres,"AP-7",km);
        if (T>=0)
        {
            int Pos=RedEspana[T].BuscarMovil(NumeMovil);
            if (Pos>=0)
            {
                  Registrado=true;
                  Terminal Movil=RedEspana[T].getMovil(Pos);
                  cout << "Pk: "<<km<<" Batería: "<<Movil.Bateria<<endl;
            }
        }
    }
    if (Registrado==false)
      cout<<"El terminal con el número "<<NumeMovil<<" no se ha registrado en autovía
AP-7\n";

    return 0;
}
```