```
#include "Punto.h"
class Pais {
  private:
    Punto p;
    string pais;
    string bandera;
  public:
    Pais () {};
    Punto GetPunto() const { ... };
    string GetPais() const { ... }
    string GetBandera() const { ... }

    bool operator < (const Pais & P) const

    bool operator == (const Pais & P) const

    friend istream & operator >> (istream & is,
                                  Pais & P)

    friend ostream & operator << (ostream & os,
                                  const Pais & P)

};
```

```
#include "Pais.h"
#include <set>
class Paises {
  private:
    set<Pais> datos;
  public:
    Paises() {}
    void Insertar (const Pais& P)
    void Borrar (const Pais& P)
    class const_iterator;
    class iterator {
      private:
        set<Pais>::iterator p;
      public:
        iterator() {}
        iterator& operator++() {
        iterator& operator--() {
        bool operator==(const iterator& it) {
        bool operator!=(const iterator& it) {
        const Pais& operator*() const {
        friend class Paises;
};
```

iterator begin()
{
}

iterator end()
{
}

iterator find(const Pair & P) {
}

friend istream & operator >> (istream & is,
Paires & R)
{
}

friend ostream & operator << (ostream & os,
const Paires & R)
{
}

```
#include "ImagenES.h"
enum Tipo_Pegado {OPACO, BLENDING};

struct Pixel {
    unsigned char r, g, b;
    unsigned char transp; //0 no 255 si
};

Class Imagen {
private:
    Pixel ** data;
    int nf, nc;
    void Borrar();
    void Copiar(const Imagen & I);

public:
    Imagen();
    Imagen(int f, int c);
    Imagen(const Imagen & I);
    Imagen & operator=(const Imagen & I);
   ~Imagen();
    //set y get
    Pixel & operator()(int i, int j);
    const Pixel & operator()(int i, int j)
                                const;
```

```
void EscribirImagen (const char * nombre);
void LeerImagen (const char * nombre, const
     string &nombremascara = "");

void LimpiarMarcasP();
int num_files() const;
int num_cols() const;
void PutImagen (int posi, int posj,
          const Imagen & I, TipoPegado tippegado=
                                        OPACO);
Imagen ExtraeImagen (int posi, int posj,
                     int dimi, int dimj);

};
```

```
Imagen::Imagen(int f, int c)
    {
    nf = f;
    nc = c;
    data = new Pixel*[nf];
    for (int i=0; i<nf; i++)
        {
        data[i] = new Pixel[nc];
        for (int j=0; j<nc; j++)
            {
            data[i][j].r = 255;
            data[i][j].g = 255;
            data[i][j].b = 255;
            data[i][j].hausp = 255;
            }
        }
    }

Pixel & Imagen::operator() (int i, int j)
    {
    assert (i>=0 && i<nf && j>=0 &&
                                j<nc*3);
    return data[i][j];
    }
```

```
void Imagen::LeerImagen(const char * nombre,
                        const string & nombremascara)
{
  int f, c;
  unsigned char * aux, *aux_mascl;
  LeerTipoImagen(nombre, f, c);
  aux = new unsigned char[f*c*3];
  LeerImagenPPM(nombre, f, c, aux);
  if (nombremascara != "")
      {
        int fm, cm;
      LeerTipoImagen(nombremascara.c_str(), fm, cm);
      aux_mascl = new unsigned char[fm*cm];
      LeerImagenPGM(nombremascara.c_str(),
                    fm, cm, aux_mascl);
      }
  else aux_mascl = 0;
}
Imagen I(f, c);
int total = f*c*3;
for (int i=0; i<total; i+=3)
      {
        int posi = i/(c*3);
        int posj = (i%(c*3))/3;
```

```
I.data[posi][posj].r = aux[i];
I.data[posi][posj].g = aux[i+1];
I.data[posi][posj].b = aux[i+2];
if (aux_mask != 0)
    I.data[posi][posj].transp = aux_mask[i/3];
else
    I.data[posi][posj].transp = 255;

*this = I;
if (aux_mask != 0) delete[] aux_mask;
delete[] aux;
}
```

```
void Imagen::PutImagen(int posi, int posj,
                       const Imagen &I, Tipo_Pegado tippegado)
{
  for(int i=0; i<I.nf; i++)
    for(int j=0; j<I.nc; j++)
      if(i+posi>=0 && i+posi<nf &&
         j+posj>=0 && j+posj<nc)
        {
          if(I.data[i][j] transp!=0)
            {
              if(tippegado==OPACO)
                data[i+posi][j+posj]=
                  =I.data[i][j];
              else {
                data[i+posi][j+posj].r = (data[i+posi][j+posi].r
                                          + I.data[i][j].r/2;
                (análogo con .b
                          .g
              }
        }
}
```