```
Public class PARCIAL {                    // UN CAMINO PASADO X LISTA DE CIUDADES.

Public ListaGenerica<String>  Resolver(Grafo<String> CIUDADES,
    String Origen, String Destino, Lista<String> PasadoPor){
  -- ListaGenerica<String> CAMINO = new ListaEnlazadaGenerica();
  --if(!CIUDADES.esVacio()){
      int pos = BuscarOyD(CIUDADES.ListaDeVertices(), Origen, Destino)
      if(pos!=-1){
        Boolean[] marca = new Boolean[cIndices.listadevertices().tamaño()+1]();
        int[] v_pasadoPor = new int[marca.length()]
        MarcarPasadoPor(v_pasadoPor, PasadoPor, CIUDADES.ListaDeVertices());

        DFS(pos, CIUDADES, Destino, marca, v_pasadoPor, CAMINO, PasadoPor.
                                                          Tamaño())
      }
  }
  return camino;
}

Private int BuscarOyD(ListaGenerica<Vertice<String>> lista, String or, String dest){
  int Origen=-1, Destino=-1;
  Lista.comenzar();
  while((Origen==-1 || Destino==-1) && (!Lista.fin())){
      Vertice<String> v = Lista.proximo();
      if((Origen==-1) && (v.dato().equals(or))){
          Origen = v.posición();
      }else if((Destino==-1) && (v.dato().equals(dest))){
          Destino = v.posición();
  }
  if(Destino==-1){
      Origen=-1;
  }
return Origen;
```

```
PUBLIC VOID MARCARPASADOPOR (INT[] MARCA, LISTAGENERICA<STRING> CIUDADES,
                             LISTAGENERICA<VERTICE<STRING>> LISTA_V);
    LISTA_V.COMENZAR();
    WHILE(!LISTA_V.FIN()){
        VERTICE<STRING> V = LISTA_V.PROXIMO().
        IF(CIUDADES.INCLUYE(V.DATO()){
            MARCA[V.POSICION()] = 1
        }
    }
}

PRIVATE BOOLEAN DFS (INT POS, GRAFO<STRING> G, STRING DESTINO, BOOLEAN[] MARCA,
        INT[] X VISITAR, LISTAGENERICA<STRING> CAMINO, INT VISITADOS){
    MARCA[POS] = TRUE; BOOLEAN CUMPLE = FALSE;
    VERTICE<STRING> V = G.VERTICE(POS);
    CAMINO.AGREGARFINAL(V.DATO());
    VISITADOS -= XVISITAR[POS];
    IF((V.DATO.EQUALS(DESTINO){
      IF(VISITADOS == 0){
       CUMPLE = TRUE;
} ELSE {
    LISTAGENERICA<ARISTA<STRING>> ADY = G.LISTADEADYACENTES(V);
    ADY.COMENZAR()
    WHILE((!CUMPLE) && (!ADY.FIN())){
        ARISTA<STRING> A = ADY.PROXIMO();
        INT J = A.VERTICEDESTINO().POSICION();
        IF[!MARCA[J]]{
CUMPLE = DFS(J, G, DESTINO, MARCA, XVISITAR, CAMINO, VISITADOS);
        }
    }
IF(!ENCONTRE){
  MARCA[POS] = FALSE;
CAMINO.ELIMINAREN(CAMINO.TAMANIO())
}
RETURN CUMPLE;
```

```
Boolean DFS (int pos, Grafo<String> g, String destino, Boolean[] marca, Lg<String> camino, Lg<String> listaCiudades);
marca[pos] = true; Boolean cumple = false;
Vertice<String> v = g.vertice(pos);
camino.agregarFinal(v.dato());
if(v.dato().equals(destino)){
    Boolean cumple = true;
    listaCiudades.comenzar()
    while(cumple && !listaCiudades.fin()){
        cumple = camino.incluye(listaCiudades.proximo());
    }
}else{
    Lista generica<Arista<String>> ady = g.listaDeAdyacentes(v);
    ady.comenzar();
    while(!encontre && !ady.fin()){
        Arista<String> a = ady.proximo();
        int j = a.verticeDestino().posicion();
        if(!marca[j]){
            cumple = DFS(j, ciudades, destino, marca, camino, listaCiudades);
        }
    }
    if(!cumple)
        marca[pos] = false;
        camino.eliminar en (camino.tamanio());
}
return cumple;
```

OPCION 2: Del programa, en este caso una vez q llegamos a destino verificamos si nuestro camino pasa por las ciudades obligatorias.