```
namespace DelegadosResumen
{
    public class Country
    {
        private List<City> _citiesList = new List<City>();

        public void CreateCitiesList()
        {
            City c1 = new City("Alicante", 350000);
            City c2 = new City("Valencia", 850000);
            City c3 = new City("Madrid", 1250000);
            City c4 = new City("Sevilla", 950000);
            City c5 = new City("Barcelona", 1100000);
            City c6 = new City("Bilbao", 870000);
            City c7 = new City("Pontevedra", 250000);

            _citiesList = new List<City>() { c1, c2, c3, c4, c5, c6,
              c7 };
        }

        // Podemos implementar una funcion 'FILTER' que ordena nuestra
          coleccion de datos
        // según una 'FUNCION DELEGADA' que introduce el usuario a
          traves de una 'LAMBDA'

        // public List Filter(FUNCION LAMBDA)
        // la funcion 'Filter' devuelve una lista segun los criterios
          de la 'FUNCION LAMBDA.


        #region Delegate Filter
        // CREAMOS UN CODIGO FUENTE QUE:
        // FILTRA ->
        // UNA COLECCION EXISTENTE MEDIANTE UNA FUNCION FILTER ->
        // A TRAVES DE UNA FUNCION DELEGADA
        // ======================================================

        // A. Definicion de la FUNCION DELEGADA:
        // La funcion NO RECIBE PARAMETROS y devuelve un BOOL.
        public delegate bool DelegateFilterPlain();


        // B. Definicion de la funcion que FILTRA:
        // Si la funcion delegada devuelve 'TRUE' para un elemento, lo
          añade a la lista resultado.
        public List<City> Filter(DelegateFilterPlain filter)
        {
            List<City> result = new List<City>();
            for (int i = 0; i < _citiesList.Count; i++)
            {
                if (filter())
                {
                    result.Add(_citiesList[i]);
```



```
                }
            }
            return result;
        }


        // C. Definicion de los parametros de la FUNCION DELEGADA:
        // Existen distintas sintaxis para la funcion filter.
        public void FilterTest()
        {
            Country countryTest = new Country();
            List<City> listResult = new List<City>();

            // sintaxis 1
            DelegateFilterPlain delegado = delegate () { return
              true; };
            DelegateFilterPlain filter = new DelegateFilterPlain
              (delegado);
            listResult = countryTest.Filter(filter);

            // sintaxis 2
            DelegateFilterPlain filter2 = new DelegateFilterPlain(() =>
               { return true; });
            listResult = countryTest.Filter(filter2);

            // sintaxis 3
            DelegateFilterPlain filter3 = new DelegateFilterPlain(() =>
               true );
            listResult = countryTest.Filter(filter3);
        }


        // A'. Definicion de la FUNCION DELEGADA:
        // La funcion RECIBE UN INT y devuelve un BOOL.
        public delegate bool DelegateFilterWithInt(int population);
        public List<City> Filter(DelegateFilterWithInt filter)
        {
            List<City> result = new List<City>();
            for (int i = 0; i < _citiesList.Count; i++)
            {
                if (filter(_citiesList[i].Population))
                {
                    result.Add(_citiesList[i]);
                }
            }
            return result;
        }


        // A''. Definicion de la FUNCION DELEGADA:
        // La funcion RECIBE UN STRING y devuelve un BOOL.
        public delegate bool DelegateFilterWithString(string name);
        public List<City> Filter(DelegateFilterWithString filter)
```

```
 98          {
 99              List<City> result = new List<City>();
100              for (int i = 0; i < _citiesList.Count; i++)
101              {
102                  if (filter(_citiesList[i].Name))
103                  {
104                      result.Add(_citiesList[i]);
105                  }
106              }
107              return result;
108          }
109          #endregion
110
111
112          public delegate int DelegateSort(City c1, City c2);
113          public List<City> Sort(DelegateSort comparator)
114          {
115              for (int i = 0; i < _citiesList.Count - 1; i++)
116              {
117                  for (int j = i + 1; j < _citiesList.Count; j++)
118                  {
119                      if (comparator(_citiesList[i], _citiesList[j]) >=
                       1)
120                      {
121                          City aux;
122                          aux = _citiesList[i];
123                          _citiesList[i] = _citiesList[j];
124                          _citiesList[j] = aux;
125                      }
126                  }
127              }
128              return _citiesList;
129          }
130
131          public delegate void DelegateVisit(City city);
132
133          public void Visit()
134          {
135              foreach (City city in _citiesList)
136              {
137
138              }
139          }
140      }
141  }
```