```csharp
namespace DelegadosConclusion
{
    internal class Program
    {
        static void Main(string[] args)
        {
            // TEST FILTER

            FilterClass<string> test1 = new FilterClass<string>();

            FilterClass<string>.DelegateFilter del1;
            del1 = new FilterClass<string>.DelegateFilter(element =>
              element.Length > 0);

            test1.Filter(del1);
            test1.Filter(element => element.Length > 0);
            test1.Filter(element => { return element.Length > 0; });
            test1.Filter(element =>
            {
                return element.Length > 0;
            });


            // TEST SORT

            SortClass<string> test2 = new SortClass<string>();

            SortClass<string>.DelegateSort del2;
            del2 = new SortClass<string>.DelegateSort((item1, item2) =>
              item1.Length < item2.Length ? 1 : 0);

            test2.Sort(del2);
            test2.Sort((item1, item2) => item1.Length < item2.Length ?
              1 : 0);
            test2.Sort((item1, item2) => { return item1.Length <
              item2.Length ? 1 : 0; });
            test2.Sort((item1, item2) =>
            {
                return item1.Length < item2.Length ? 1 : 0;
            });


            // TEST VISIT

            List<string> list = new List<string>();

            VisitClass<string> test3 = new VisitClass<string>();

            VisitClass<string>.DelegateVisit del3;
            del3 = new VisitClass<string>.DelegateVisit(element =>
              element = "Elemento: " + element);

            test3.Visit(del3);
            test3.Visit(element => element = "Elemento: " + element);
            test3.Visit(element => {element = "Elemento: " + element;});
            test3.Visit(element =>
            {
                element = "Elemento: " + element;
            });
        }
    }
}
```

```csharp
namespace DelegadosConclusion
{

    public class FilterClass<T>
    {
        private List<T> _list = new List<T>();

        public delegate bool DelegateFilter(T element);

        public List<T> Filter(DelegateFilter del)
        {
            if (del == null)
                return null;

            List<T> filterList = new List<T>();

            for (int i = 0; i < _list.Count; i++)
            {
                if (del(_list[i]))
                    filterList.Add(_list[i]);
            }

            return filterList;
        }
    }
}
```

```
namespace DelegadosConclusion
{
    public class SortClass<T>
    {
        private List<T> _list = new List<T>();

        public delegate int DelegateSort(T element1, T element2);

        public List<T> Sort(DelegateSort comparison)
        {
            if (comparison == null)
                return null;

            List<T> listSort = new List<T>();

            for(int i = 0; i < _list.Count - 1; i++)
            {
                for(int j = i + 1; i < _list.Count; j++)
                {
                    if (comparison(_list[i], _list[j]) >= 1)
                    {
                        T aux;
                        aux = _list[i];
                        _list[j] = _list[i];
                        _list[j] = aux;
                    }
                }
            }
            return listSort;
        }
    }
}
```

```
namespace DelegadosConclusion
{
    public class VisitClass<T>
    {
        private List<T> _list = new List<T>();

        public delegate void DelegateVisit(T visitor);

        public void Visit(DelegateVisit visitor)
        {
            if (visitor == null)
                return;

            foreach(T element in _list)
            {
                visitor(element);
            }
        }
    }
}

```