```
//Created by Andrew Owen for Professor Vanselow
//This code is based on a LINQ walk-through


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LinqToNorthwindCmd
{
    class Customer
    {
        public string CustomerID { get; set; }
        public string City { get; set; }

        public override string ToString()
        {
            return CustomerID + "\t" + City;
        }
    }
}
```

```
//Created by Andrew Owen for Professor Vanselow
//This code sample demonstrates the power of LINQ
//And is based on a LINQ walk-through

using System;
using System.Collections.Generic;
using System.Linq;
using System.Xml.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.SqlClient;

namespace LinqToNorthwindCmd
{

    class Program
    {
        static void Main(string[] args)
        {
            Console.SetBufferSize(100, 1000);
            Console.WriteLine("This program is written for the purposes of" +
                "of demonstrating the use of LINQ");

            NumQuery();
            ObjectQuery();
            XMLQuery();
            XMLQueryWithoutObject();
            XMLTransformQuery();
        }

        static IEnumerable<Customer> CreateCustomers()
        {
            return new List<Customer> {
                new Customer { CustomerID = "ALFKI", City = "Berlin"    },
                new Customer { CustomerID = "BONAP", City = "Marseille" },
                new Customer { CustomerID = "CONSH", City = "London"    },
                new Customer { CustomerID = "EASTC", City = "London"    },
                new Customer { CustomerID = "FRANS", City = "Torino"    },
                new Customer { CustomerID = "LONEP", City = "Portland"  },
                new Customer { CustomerID = "NORTS", City = "London"    },
                new Customer { CustomerID = "THEBI", City = "Portland"  }
            };
        }

        static IEnumerable<XElement> getCustomersXML()
        {
            return from c in XDocument.Load("Customers.xml")
                                .Descendants( "Customers").Descendants()
                   select c;
        }

        static IEnumerable<Customer> CreateCustomersFromXML()
        {
            return from c in getCustomersXML()
                   select new Customer
                   {
                       City = c.Attribute( "City").Value,
                       CustomerID = c.Attribute( "CustomerID").Value
                   };
        }

        static void waitForInput()
        {
            Console.Write("\nPress any key to continue...");
            Console.ReadLine();
            Console.Clear();
        }

        static string GetHeadersForCustomers(params int[] numHeaders)
        {
            //Get the headers from getCustomersXML
            //By joining all the attribute names into a string
            //Taking the number indicated by numHeaders
            var str =
                from x in getCustomersXML()
                select
                    String.Join("\t",
                    x.Attributes()
                    .Where((y, idx) => numHeaders.Contains(idx))
                    .Select(att => att.Name));
            return str.FirstOrDefault();
        }

        static void XMLQuery()
        {
            Console.WriteLine("Translating XML to objects\n");

            Console.WriteLine(GetHeadersForCustomers(0,1));
            var customers = CreateCustomersFromXML()
                .Where(c => c.City ==  "London");
            foreach (var c in customers)
            {
                Console.WriteLine(c.CustomerID.PadRight(16) + c.City);
            }
            waitForInput();
        }

        static void ObjectQuery()
        {
            Console.WriteLine("Objects Created in code\n");
            var results = from c in CreateCustomers()
                          where c.City == "London"
                          select c;
            foreach (var c in results)
            {
                Console.WriteLine(c);
            }
            waitForInput();
        }
        public static void XMLQueryWithoutObject()
        {
            Console.WriteLine("XML from file\n");
            var doc = XDocument.Load("Customers.xml");

            var results = from c in doc.Descendants("Customer")
                          where c.Attribute("City").Value == "London"
                          select c;
            Console.WriteLine("Results:\n");
            foreach (var contact in results)
            {
                Console.WriteLine("{0}", contact);
            }
            waitForInput();
        }

        public static void XMLTransformQuery()
        {
            Console.WriteLine("XML filer and transformation\n");
            XDocument doc = XDocument.Load("Customers.xml");

            Console.WriteLine("Input XML\n");
            Console.WriteLine(doc);
            waitForInput();
            Console.WriteLine("XML filter and transformation\n");

            var results =
                from c in doc.Descendants("Customer")
                where c.Attribute("City").Value == "London"
                select c;

            XElement transformedResults =
                new XElement("Londoners",
                    from customer in results
                    select new XElement("Contact",
                        new XAttribute("ID",
                            customer.Attribute( "CustomerID").Value),
                        new XElement("Name",
                            customer.Attribute( "ContactName").Value),
                        new XElement("City",
                            customer.Attribute( "City").Value)
                        )
                    );

            Console.WriteLine("Output:\n{0}", transformedResults);
            transformedResults.Save( "Output.xml");
            waitForInput();

        }

        //Linq over primitives
        static void NumQuery()
        {
            var numbers = new int[] { 1, 4, 9, 16, 25, 36 };
            var evenNumbers = numbers.Where(x => x % 2 == 0);
            Console.WriteLine("Even square numbers:\n");
            foreach (var item in evenNumbers)
            {
                Console.WriteLine(item);
            }
            waitForInput();
        }
    }
}
```