\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**CSC253 ADVANCED C# ProGRAMMING**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

LAB 04 **LINQ**

# Objectives

In this lab assignment, students will learn:

* To use LINQ to query arrays
* To use LINQ to query lists
* To use LINQ to select items in a query
* To use LINQ to sort items in a query

# Goals

In this lab assignment, students will demonstrate the abilities to:

* Use LINQ to query arrays
* Use LINQ to query lists
* Use LINQ to select items in a query
* Use LINQ to sort items in a query

# Description

Create a C# console application for each question. When you create a new C# project, Visual Studio creates a folder to hold every file and sub-folder for your project. You need to zip this folder and submit the zip file to Blackboard.

1. [60 points] The following is class Invoice, which includes four properties: PartNumber (type string), PartDescription (type string), Quantity (type int) and UnitPrice (type decimal). Class Invoice also has a GetInvoiceAmout method to calculate invoice total, and a ToString method to display the four properties.

// Invoice.cs

// Invoice for a hardware company

using System;

public class Invoice

{

string partNum;

string partDescription;

int quantity;

decimal unitPrice;

public Invoice(string pNum, string pDesciption, int q, decimal uPrice)

{

PartNum = pNum;

PartDescription = pDesciption;

Quantity = q;

UnitPrice = uPrice;

}

public string PartNum

{

get

{

return partNum;

} // end get

set

{

partNum = value;

} // end set

} // end

public string PartDescription

{

get

{

return partDescription;

} // end get

set

{

partDescription = value;

} // end set

} // end

public int Quantity

{

get

{

return quantity;

} // end get

set

{

if (value >= 0)

quantity = value;

} // end set

} // end

public decimal UnitPrice

{

get

{

return unitPrice;

} // end get

set

{

if (value >= 0M)

unitPrice = value;

} // end set

} // end

public decimal GetInvoiceAmount()

{

decimal amt = Quantity \* UnitPrice;

return amt;

}

public override string ToString()

{

return string.Format("{0,-10}{1,-20}{2,-10}{3,10:C}", PartNum,

PartDescription, Quantity, UnitPrice);

}

}

Write a C# application to do the following.

First, create an array of Invoice objects with the following data:

|  |  |  |  |
| --- | --- | --- | --- |
| Part number | Part description | Quantity | Price |
| 83 | Electric sander | 7 | 57.98 |
| 24 | Power saw | 18 | 99.99 |
| 7 | Sledge hammer | 11 | 21.50 |
| 77 | Hammer | 76 | 11.99 |
| 39 | Lawn mower | 3 | 79.50 |
| 68 | Screwdriver | 106 | 6.99 |
| 56 | Jig saw | 21 | 11.00 |
| 3 | Wrench | 34 | 7.50 |

Then perform the following queries on the array of Invoice objects and display the results:

(a) Use LINQ to sort the Invoice objects by PartDescription.

(b) Use LINQ to sort the Invoice objects by UnitPrice.

(c) Use LINQ to select the PartDescription and Quantity and sort the results by Quantity.

(d) Use LINQ to select from each invoice the PartDescription and the invoice total amount. The GetInvoiceAmount method returns the total amount of the invoice. Order the results by invoice total amount. [Hint: Use let in the LINQ query to store invoice total amount in a new variable. See the example in textbook Figure 9.7]

(e) Using the results of the LINQ query in part (d), select invoices with total amount in the range $200 to $500.

The output of the program should look like the following:

Invoices sorted by part description

83 Electric sander 7 $57.98

77 Hammer 76 $11.99

56 Jig saw 21 $11.00

39 Lawn mower 3 $79.50

24 Power saw 18 $99.99

68 Screwdriver 106 $6.99

7 Sledge hammer 11 $21.50

3 Wrench 34 $7.50

Invoices sorted by unit price

68 Screwdriver 106 $6.99

3 Wrench 34 $7.50

56 Jig saw 21 $11.00

77 Hammer 76 $11.99

7 Sledge hammer 11 $21.50

83 Electric sander 7 $57.98

39 Lawn mower 3 $79.50

24 Power saw 18 $99.99

Part description and quantity sorted by quantity

{ PartDescription = Lawn mower, Quantity = 3 }

{ PartDescription = Electric sander, Quantity = 7 }

{ PartDescription = Sledge hammer, Quantity = 11 }

{ PartDescription = Power saw, Quantity = 18 }

{ PartDescription = Jig saw, Quantity = 21 }

{ PartDescription = Wrench, Quantity = 34 }

{ PartDescription = Hammer, Quantity = 76 }

{ PartDescription = Screwdriver, Quantity = 106 }

Part description and invoice total sorted by invoice total

{ PartDescription = Jig saw, invoiceTotal = 231.00 }

{ PartDescription = Sledge hammer, invoiceTotal = 236.50 }

{ PartDescription = Lawn mower, invoiceTotal = 238.50 }

{ PartDescription = Wrench, invoiceTotal = 255.00 }

{ PartDescription = Electric sander, invoiceTotal = 405.86 }

{ PartDescription = Screwdriver, invoiceTotal = 740.94 }

{ PartDescription = Hammer, invoiceTotal = 911.24 }

{ PartDescription = Power saw, invoiceTotal = 1799.82 }

Part description and invoice total, total between $200 and $500

{ PartDescription = Jig saw, invoiceTotal = 231.00 }

{ PartDescription = Sledge hammer, invoiceTotal = 236.50 }

{ PartDescription = Lawn mower, invoiceTotal = 238.50 }

{ PartDescription = Wrench, invoiceTotal = 255.00 }

{ PartDescription = Electric sander, invoiceTotal = 405.86 }

2. [40 points] Write a console application that inserts 30 random integers in the range 1 to 20 into a List<int>. Perform the following queries on the list and display your results:

(a) Use LINQ to sort the list in ascending order. Display the list.

(b) Use LINQ to sort the list in descending order. Display the list.

(c) Display the list in ascending order with duplicates removed.

# Grading rubric for Program 1

Sort by part description [12 pts]

Sort by unit price [12 pts]

Sort by part description [12 pts]

Select by part description and invoice total amount [12 pts]

Select invoices with total amount $200 - $500 [12 pts]

# Grading rubric for Program 2

Create list of random numbers [10 pts]

Sort list in ascending order [10 pts]

Sort list in descending order [10 pts]

Display list sorted with no duplicates [10 pts]