

Good project with a couple  
minor problems.

**CISP 41**

# Programming in C#

## Project Evaluation Sheet

Student Name: \_\_\_\_\_ Bryant Tunbutr \_\_\_\_\_ Project Number: \_\_\_\_\_ 2 \_\_\_\_\_

Project Name: \_\_\_\_\_ BtunbutrProject2 \_\_\_\_\_ Visual Studio Version: \_\_\_\_\_ 2008 \_\_\_\_\_

Date Due: \_\_\_\_\_ 10/11/12 \_\_\_\_\_ Date Turned In: \_\_\_\_\_ 10/11/12 \_\_\_\_\_

Above to be completed by student

**Points ( 40 Possible)**

### Correctness/Efficiency:

Output is accurate \_\_\_\_\_

Meets all requirements \_\_\_\_\_ -2 \_\_\_\_\_

Provide appropriate user interface \_\_\_\_\_

Logic is efficient \_\_\_\_\_

### Documentation/Coding Style:

Project can be open from the submitted zip file \_\_\_\_\_

Folder is present and contains all necessary project files (no extra files) \_\_\_\_\_

Use required coding template \_\_\_\_\_

Use proper naming and spacing \_\_\_\_\_

Submit all requested information \_\_\_\_\_

### Test Cases:

List all required test cases \_\_\_\_\_

Provide output forms for important test cases \_\_\_\_\_

### Other issues:

\_\_\_\_\_

### Extra Credit:

\_\_\_\_\_ +4 \_\_\_\_\_

### Timeliness:

\_\_\_\_\_

### Project Score:

42 / 40

### Project specification

---

This software is intended to summarize, calculate, and display costs for personalized shirts for the company Cool Boards. It is designed to be run in Visual Studio 2008 using the C# coding. It uses user input including the shirt size, quantity, additional features like monograms or pockets.

It displays total cost, summary, and provides information about the program with an about button.

Project status

---

The project is completed and finalized. Extra credit summary and order complete features are included.

Sketch of user interface

---

Provided on page 216 of textbook

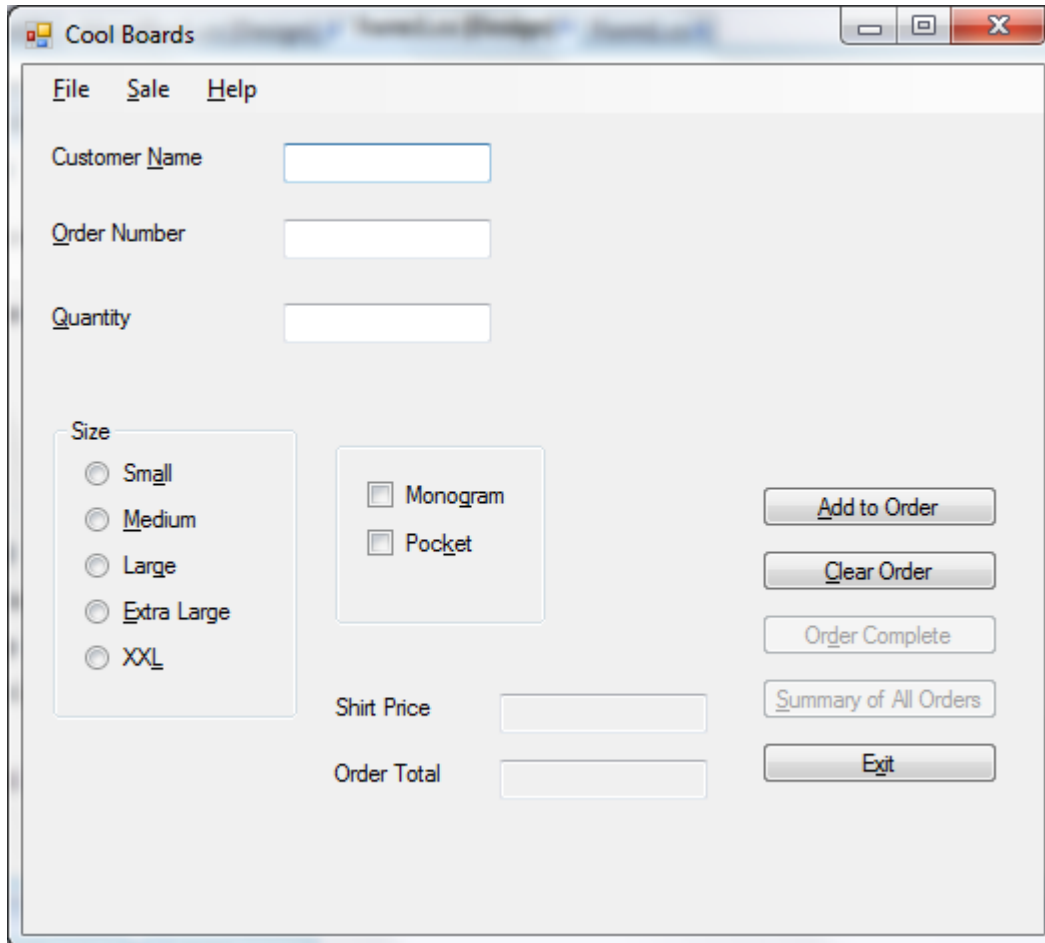
[illegible]

[illegible]

## Test cases and captured screens

## Test case #1

Ordering 10 small shirts with pockets



The screenshot shows a Windows application window titled "Cool Boards". The window has a menu bar with "File", "Sale", and "Help". Below the menu bar, there are three input fields: "Customer Name", "Order Number", and "Quantity". To the left of these fields is a "Size" section with five radio buttons: "Small", "Medium", "Large", "Extra Large", and "XXL". To the right of the "Size" section are two checkboxes: "Monogram" and "Pocket". Below these checkboxes are two input fields: "Shirt Price" and "Order Total". On the right side of the window, there are five buttons: "Add to Order", "Clear Order", "Order Complete", "Summary of All Orders", and "Exit".

File Sale Help

Customer Name

Order Number

Quantity

Size

☐ Small

☐ Medium

☐ Large

☐ Extra Large

☐ XXL

☐ Monogram

☐ Pocket

Shirt Price

Order Total

Add to Order

Clear Order

Order Complete

Summary of All Orders

Exit

Cool Boards

File Sale Help

Customer Name

Order Number

Quantity

Size

☒ Small

☐ Medium

☐ Large

☐ Extra Large

☐ XXL

☐ Monogram

☒ Pocket


Shirt Price

Order Total

Order Complete

Amount Due \$110.00

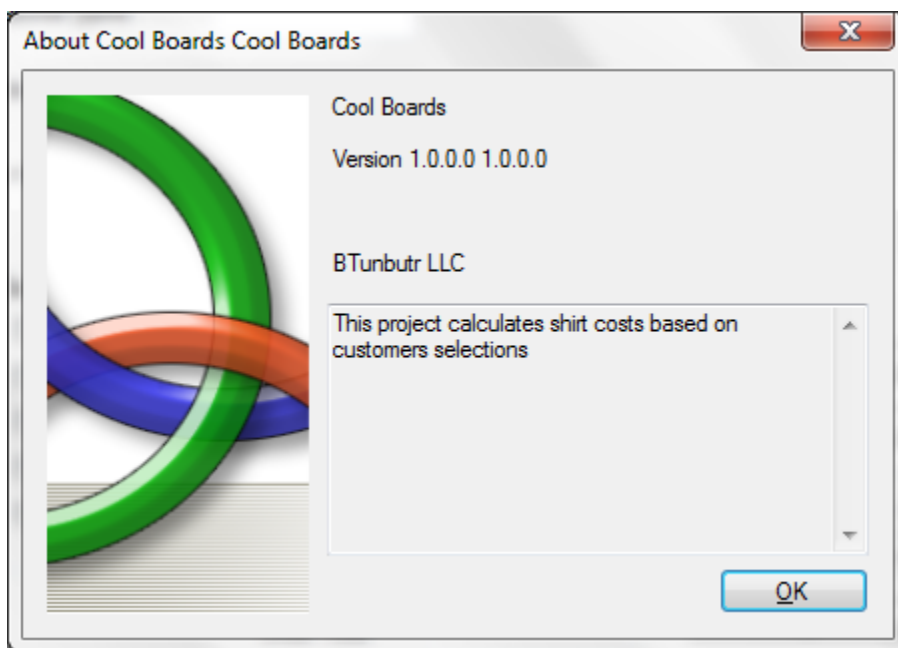
Shirt Sales Summary

 Shirts Sold: 10

Number of Orders: 1

Total Sales: \$110.00





**Test case #2**

**Ordering 20 large shirts, then adding 20 more large shirts with the same customer barb**

The screenshot shows a Windows-style application window titled "Cool Boards". It has a menu bar with "File", "Sale", and "Help". The main area contains several input fields and buttons. The "Customer Name" field is filled with "barb", the "Order Number" field with "bb", and the "Quantity" field with "20". Under the "Size" section, the "Large" radio button is selected. There are checkboxes for "Monogram" and "Pocket", both of which are unchecked. On the right side, there are five buttons: "Add to Order" (highlighted with a blue dotted border), "Clear Order", "Order Complete", "Summary of All Orders", and "Exit". At the bottom, there are two summary fields: "Shirt Price" showing "\$10.00" and "Order Total" showing "\$200.00".

Field	Value
Customer Name	barb
Order Number	bb
Quantity	20
Size	Large
Monogram	Unchecked
Pocket	Unchecked
Shirt Price	\$10.00
Order Total	\$200.00

Cool Boards

File Sale Help

Customer Name

Order Number

Quantity

Size

☐ Small

☐ Medium

☒ Large

☐ Extra Large

☐ XXL

☐ Monogram

☐ Pocket


Shirt Price

Order Total

Order Complete

Amount Due \$400.00

Shirt Sales Summary

 Shirts Sold: 40

Number of Orders: 1

Total Sales: \$400.00

**Test case #3**

**A orders 11 XXL shirts with monograms and pockets**

**B then orders 22 large shirts**

**C then orders 333 XL shirts with pockets.**

The 'Cool Boards' application window displays the following fields and controls:

- File Sale Help** (Menu bar)
- Customer Name**: Text box containing 'a'
- Order Number**: Text box containing 'a'
- Quantity**: Text box containing '11'
- Size**: Radio button group with options: Small, Medium, Large, Extra Large, and **XXL** (selected).
- Monogram**: Check box (checked)
- Pocket**: Check box (checked)
- Buttons**: 'Add to Order' (blue), 'Clear Order' (grey), 'Order Complete' (grey), 'Summary of All Orders' (grey), and 'Exit' (blue).
- Shirt Price**: Text box showing '\$15.00'
- Order Total**: Text box showing '\$165.00'

The 'Order Complete' dialog box displays the following information:

- Amount Due**: \$165.00
- OK** button (blue)

Cool Boards

File Sale Help

Customer Name

Order Number

Quantity

Size

☐ Small

☐ Medium

☒ Large

☐ Extra Large

☐ XXL

☐ Monogram

☐ Pocket

Shirt Price

Order Total

Order Complete

Amount Due \$220.00

Cool Boards

File Sale Help

Customer Name

Order Number

Quantity

Size

☐ Small

☐ Medium

☐ Large

☒ Extra Large

☐ XXL

☐ Monogram

☒ Pocket


Shirt Price

Order Total

Order Complete

Amount Due \$3,996.00

Shirt Sales Summary

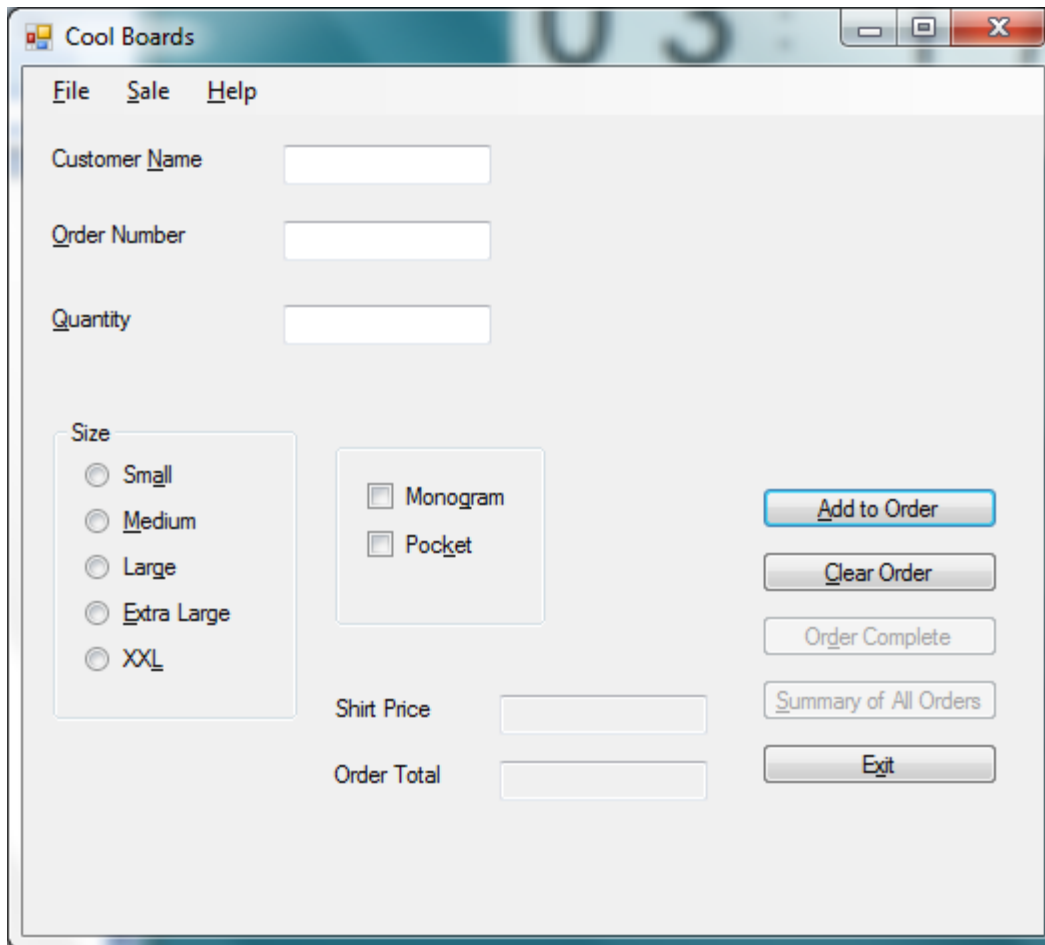
 Shirts Sold: 366

Number of Orders: 3

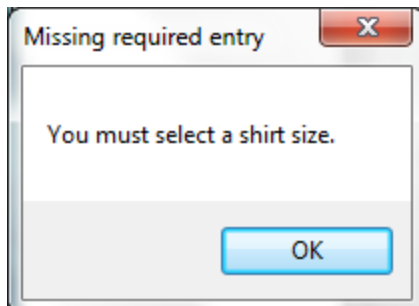
Total Sales: \$4,381.00

#### Test case #4

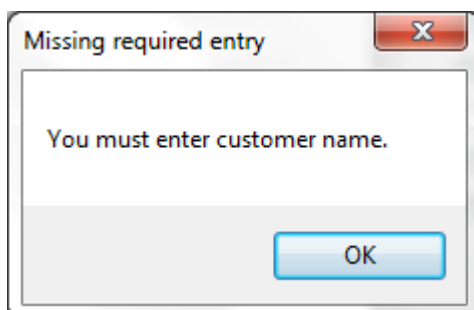
Person forgets to enter data and just clicks add to order



The 'Cool Boards' application window features a menu bar with 'File', 'Sale', and 'Help'. It contains three input fields for 'Customer Name', 'Order Number', and 'Quantity'. A 'Size' section offers radio button options for Small, Medium, Large, Extra Large, and XXL. Two checkboxes for 'Monogram' and 'Pocket' are also present. On the right, a vertical stack of buttons includes 'Add to Order' (highlighted in blue), 'Clear Order', 'Order Complete', 'Summary of All Orders', and 'Exit'. At the bottom, there are input fields for 'Shirt Price' and 'Order Total'.

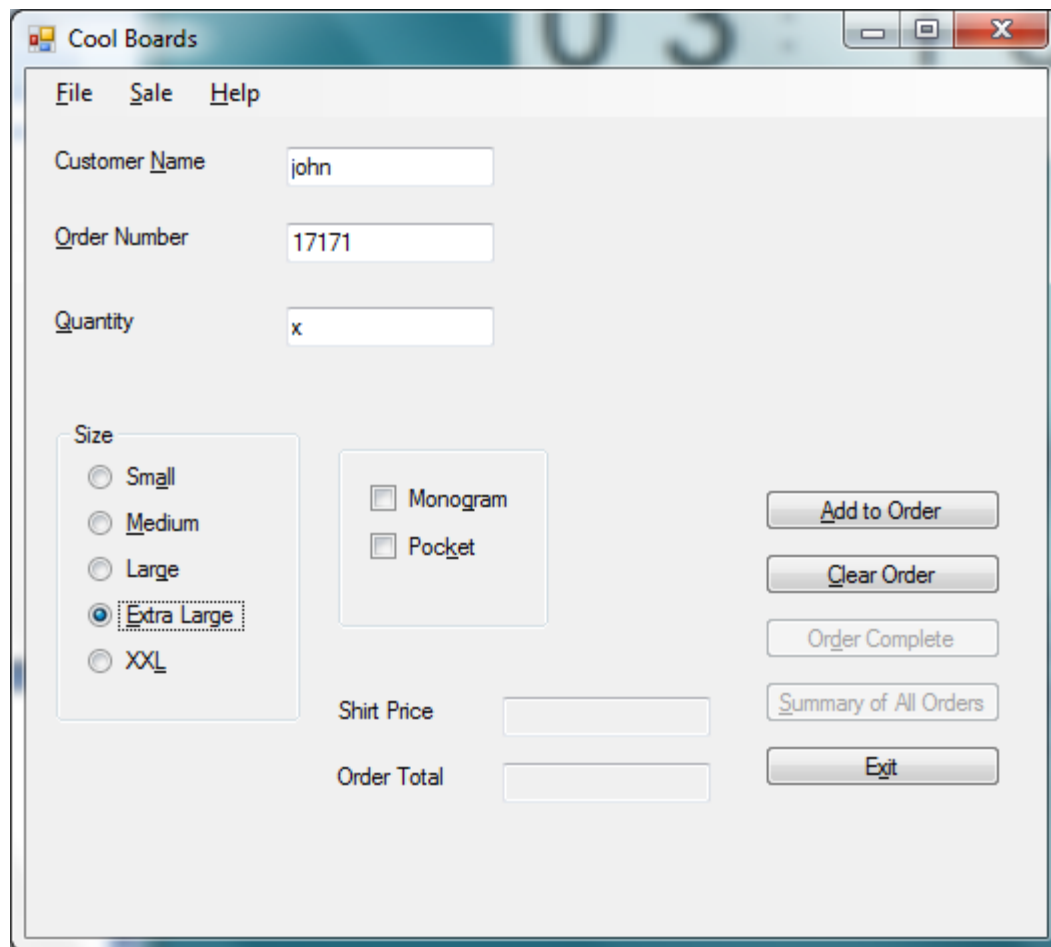


A 'Missing required entry' dialog box with a red close button. The message reads: 'You must select a shirt size.' An 'OK' button is located at the bottom right.

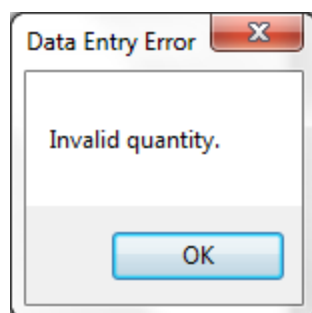


A 'Missing required entry' dialog box with a red close button. The message reads: 'You must enter customer name.' An 'OK' button is located at the bottom right.

Test case #5  
John writes x as a quantity



The image shows a screenshot of a Windows application window titled "Cool Boards". The window has a menu bar with "File", "Sale", and "Help". Below the menu bar, there are three text input fields: "Customer Name" with the value "john", "Order Number" with the value "17171", and "Quantity" with the value "x". Below these fields, there is a "Size" section with five radio buttons: "Small", "Medium", "Large", "Extra Large" (which is selected), and "XXL". To the right of the "Size" section, there are two checkboxes: "Monogram" and "Pocket". Below these checkboxes, there are two text input fields: "Shirt Price" and "Order Total". To the right of these fields, there are five buttons: "Add to Order", "Clear Order", "Order Complete", "Summary of All Orders", and "Exit".



The image shows a screenshot of a "Data Entry Error" dialog box. The dialog box has a title bar with the text "Data Entry Error" and a close button. The main area of the dialog box contains the text "Invalid quantity." Below this text, there is an "OK" button.



Source code

---

```
/*  Program:      Cool Boards
    Author:       Bryant Tunbutr
    Class:        CISP41-22726201220
    Date:         10/11/12
    Description:   This project calculates the amount due
                  based on the customer selection
                  and accumulates summary data for all customers.
                  Includes menus, common dialog boxes, and general methods.

    I certify that the code below is my own work.

    Exception(s): N/A

*/
using System;
using System.Windows.Forms;

namespace WindowsFormsApplication13
{
    public partial class Form1 : Form
    {
        // Declare class variables.
        private decimal itemPriceDecimal,
                       totalOrderDecimal,
                       shirtPriceDecimal,
                       totalSalesDecimal;
        private int shirtsInteger,
                   ordersInteger;

        public Form1()
        {
            InitializeComponent();
        }

        private void exitButton_Click(object sender, EventArgs e)
        {
            // To close program.
            this.Close();
        }

        private void addToOrderButton_Click(object sender, EventArgs e)
        {
            // Add the current item price and quantity to the order.
            int extrasInteger = 0;
            if (noSizeRadioButton.Checked)
            {
                // Error message for missing info.
                MessageBox.Show("You must select a shirt size.",
                                "Missing required entry");
            }
        }
    }
}
```

```

if (customerTextBox.Text == "")
{
    // Error message for missing info.
    MessageBox.Show("You must enter customer name.",
        "Missing required entry");
}
else
{
    try
    {
        // Calculate price based on size and quantity.
        int quantityInteger = int.Parse(quantityTextBox.Text);
        if (quantityInteger != 0)
        {
            if (smallRadioButton.Checked)
            {
                itemPriceDecimal = 10m;
            }
            if (mediumRadioButton.Checked)
            {
                itemPriceDecimal = 10m;
            }
            if (largeRadioButton.Checked)
            {
                itemPriceDecimal = 10m;
            }
            else if (extraLargeRadioButton.Checked)
            {
                itemPriceDecimal = 11m;
            }
            else if (xxlRadioButton.Checked)
            {
                itemPriceDecimal = 12m;
            }

            // Calculate price based on pockets and monograms.
            extrasInteger = 0;
            if (pocketCheckBox.Checked)
            {
                extrasInteger++;
            }
            if (monogramCheckBox.Checked)
            {
                extrasInteger+=2;
            }
            itemPriceDecimal += extrasInteger * 1m; // 1 dollar for
each item.

            // display price of one shirt.
            shirtPriceDecimal =
findshirtPriceDecimal(itemPriceDecimal, quantityInteger);
            shirtPriceTextBox.Text = itemPriceDecimal.ToString("C");

            shirtPriceTextBox.Text = itemPriceDecimal.ToString("C");

            // running total of shirts
            shirtsInteger += quantityInteger;

            // running total of orders
            totalOrderDecimal += shirtPriceDecimal;

```

```

        // total of cost of order
        orderTotalTextBox.Text =
totalOrderDecimal.ToString("C");

        customerTextBox.Enabled = false;
        orderTextBox.Enabled = false;
        orderCompleteButton.Enabled = true;

    }
    else
    {
        // Error message for missing info.
        MessageBox.Show("Please enter a quantity.",
            "Missing Required Entry");
    }

}

catch (FormatException)
{
    // Error message for bad info.
    MessageBox.Show("Invalid quantity.", "Data Entry Error");
    quantityTextBox.Focus();
    quantityTextBox.SelectAll();
}
}

private decimal findshirtPriceDecimal(decimal itemPriceDecimal, int
quantityInteger)
{
    // method to find price.
    return (itemPriceDecimal * quantityInteger);
}

private void clearOrderButton_Click(object sender, EventArgs e)
{
    // clear everything for next customer
    customerTextBox.Text = "";
    orderTextBox.Text = "";
    quantityTextBox.Text = "";
    shirtPriceTextBox.Text = "";           Need to disable "Order Complete"
    orderTotalTextBox.Text = "";
    totalOrderDecimal = 0;
    customerTextBox.Enabled = true;
    orderTextBox.Enabled = true;
    summaryOfAllOrdersButton.Enabled = false;
}

private void orderCompleteButton_Click(object sender, EventArgs e)
{
    // Order is complete, add to summary and clear order.

    // Check if the last item was added to the total.
    if (shirtPriceTextBox.Text != "")
    {
        DialogResult responseDialogResult;
        string messageString = "Current Item not recorded. Add to
order?";

        responseDialogResult = MessageBox.Show(messageString,
            "Verify Last Shirt Purchase",

```

```

        Problems here. Always ask to add to order again
        (already added). Also need to set up for next
        MessageBoxButtons.YesNo, customer.
        MessageBoxIcon.Question);
    if (responseDialogResult == DialogResult.Yes)
    {
        addToOrderButton_Click(sender, e);
    }
}

// Display amount due.
string dueString = "Amount Due " + totalOrderDecimal.ToString("C");
MessageBox.Show(dueString, "Order Complete");

// Add to summary totals.
ordersInteger++;
totalSalesDecimal += totalOrderDecimal;

// Reset buttons and total for new order.
summaryOfAllOrdersButton.Enabled = true;
orderCompleteButton.Enabled = false;
totalOrderDecimal = 0m;
}

private void summaryOfAllOrdersButton_Click(object sender, EventArgs e)
{
    // Display the summary information in a message box.

    string summaryString = "Shirts Sold:      "
        + shirtsInteger.ToString()
        + "\n\n" + "Number of Orders: "
        + ordersInteger.ToString()
        + "\n\n" + "Total Sales:      "
        + totalSalesDecimal.ToString("C");
    MessageBox.Show(summaryString, "Shirt Sales Summary",
        MessageBoxButtons.OK,
        MessageBoxIcon.Information);
}

private void aboutToolStripMenuItem_Click(object sender, EventArgs e)
{
    // show programmer info.
    AboutBox1 aboutForm = new AboutBox1();
    aboutForm.ShowDialog();
}
}
}

```