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 (____ Possible)

Correctness/Efficiency:

Output is accurate _____

Meets all requirements _____

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: _____

Timeliness: _____

Project Score:

CISP 41 Programming in C#

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.

CISP 41 **Programming in C#**

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

CISP 41 **Programming in C#**

Objects and Properties Plan for _____Form

Object	Property	Setting
exitButton	Name Text	E&xit
clearButton	Name Text	C&lear
groupBox1	Text	
label1	Name Text	**Order Total**
Radiobutton	Text	**Size**
about	Name Text	**Info**
BTunbutrProject2	Name Text	**Form**
Menu	Name Text	**Click**
Check box	Text	**Click feature shirt**

CISP 41 **Programming in C#**

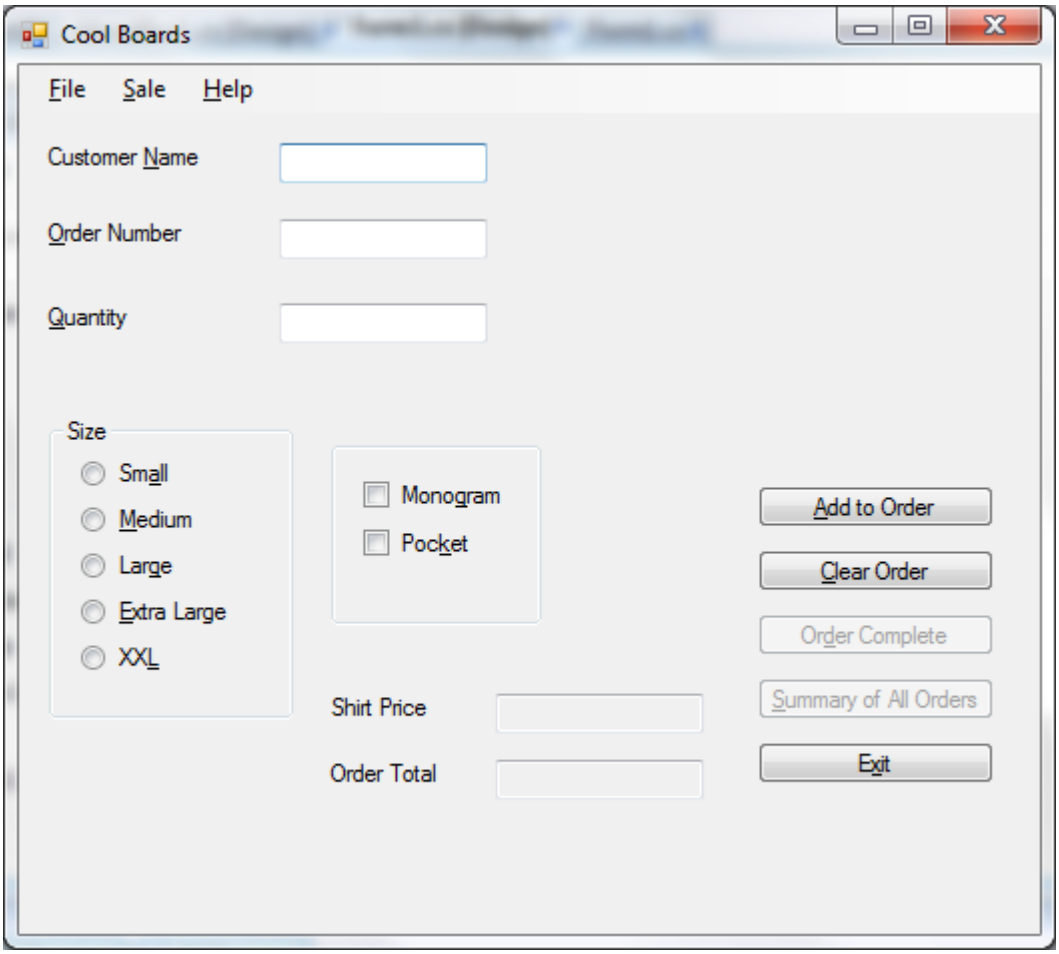
Event Plan for ___ BTunbutrProject 2_____Form

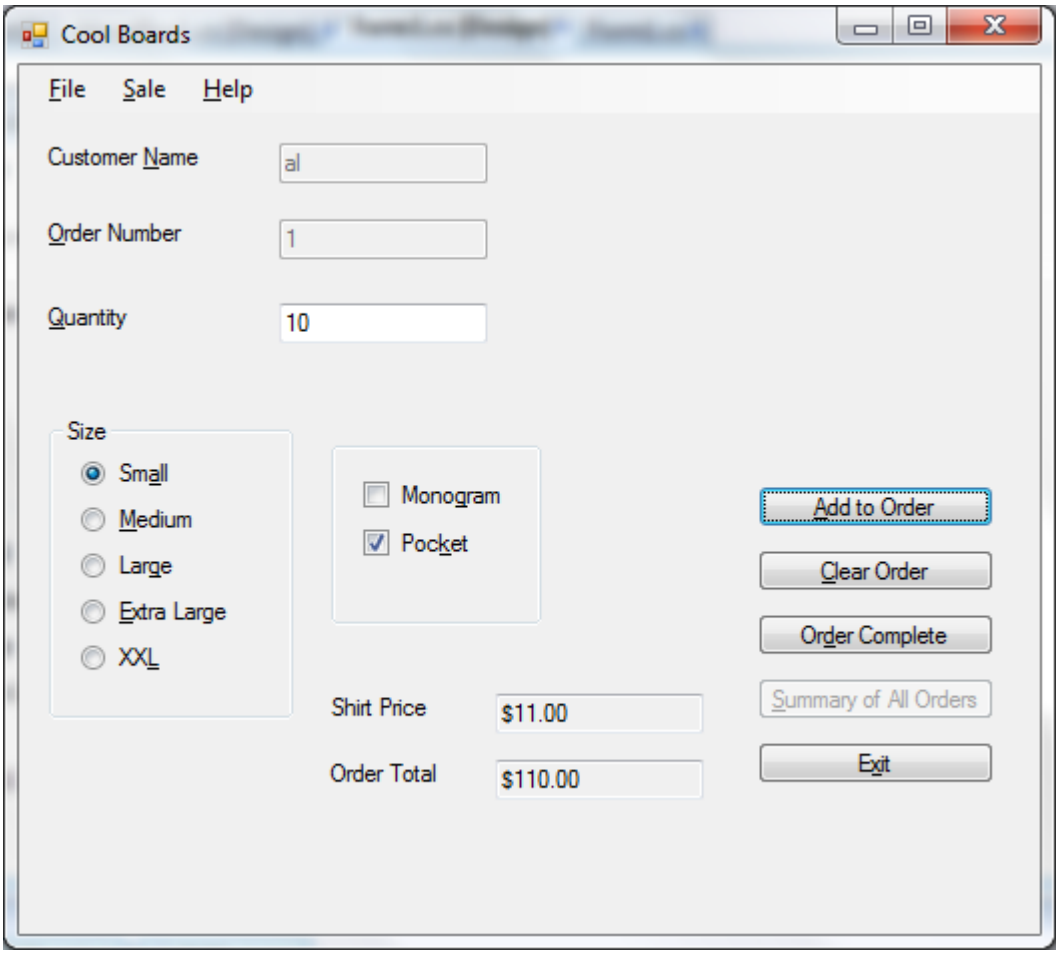
Object	Event	Action - Pseudocode
orderCompleteButton	Click	If last item not cleared from screen Ask user whether to add it. If yes Call addToOrderButton_Click. Display the price of the order. Add to the number of orders and totalSales. Reset the controls for the next order. Clear the order amount
summaryButton	Click	Display the summary totals in a message box.
aboutToolStripMenuItem	Click	Display a message box showing programmer and version
exitButton	Click	exit

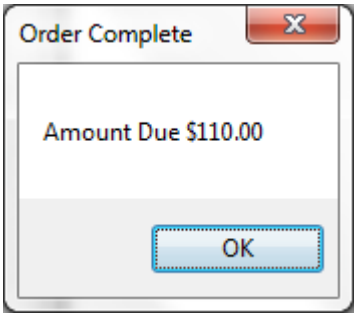
CISP 41 Programming in C#

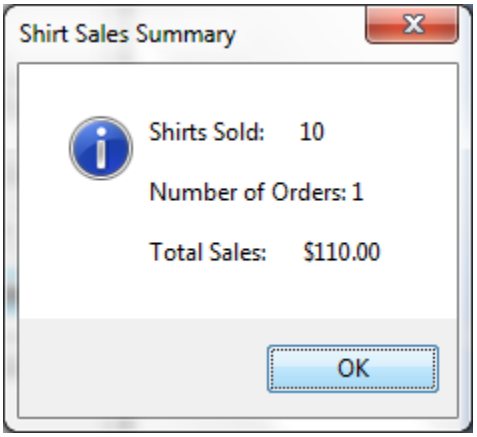
Test cases and captured screens

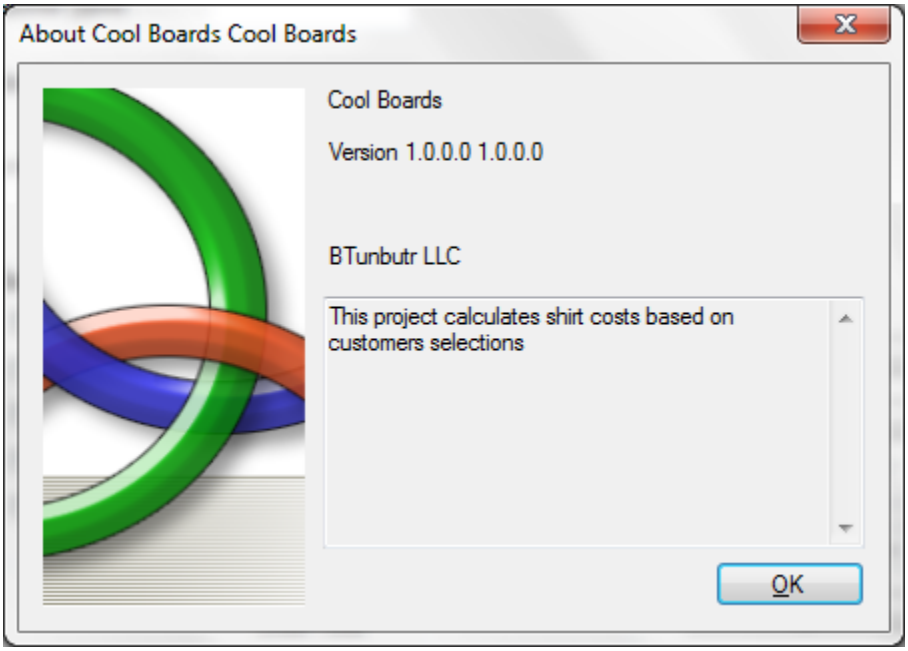
Test case #1
Ordering 10 small shirts with pockets



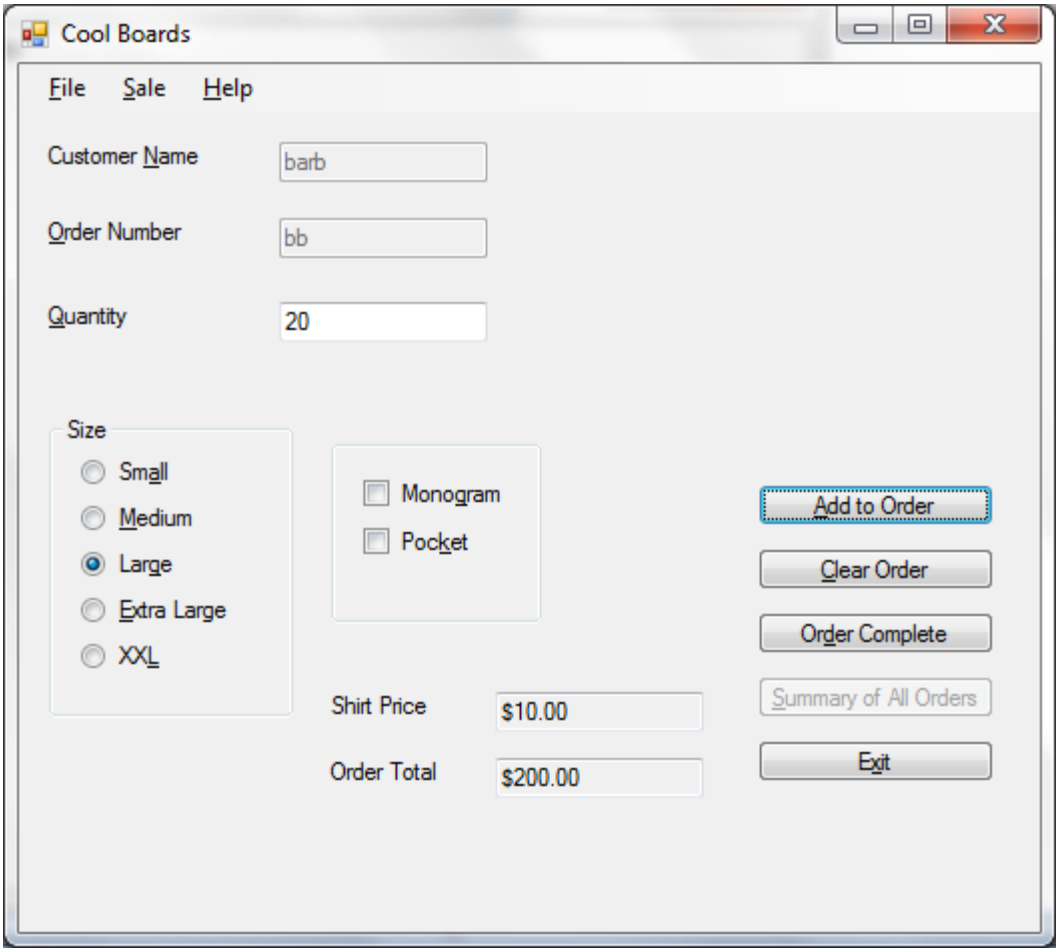

Cool Boards
File
Sale
Help
Customer Name
al
Order Number
1
Quantity
10
Size
Small
Medium
Large
Extra Large
XXL
Monogram
Pocket
Add to Order
Clear Order
Order Complete
Summary of All Orders
Exit
Shirt Price
$11.00
Order Total
$110.00


Order Complete
Amount Due $110.00
OK


Shirt Sales Summary
Shirts Sold: 10
Number of Orders: 1
Total Sales: $110.00
OK


About Cool Boards Cool Boards
Cool Boards
Version 1.0.0.0 1.0.0.0
BTunbutr LLC
This project calculates shirt costs based on customers selections
OK

Test case #2
Ordering 20 large shirts, then adding 20 more large shirts with the same customer barb



Cool Boards

File Sale Help

Customer Name barb

Order Number bb

Quantity 20

Size

- Small
- Medium
- Large
- Extra Large
- XXL

- Monogram
- Pocket

Add to Order

Clear Order

Order Complete

Summary of All Orders

Exit

Shirt Price $10.00

Order Total $400.00

Order Complete

Amount Due $400.00

OK

Shirt Sales Summary

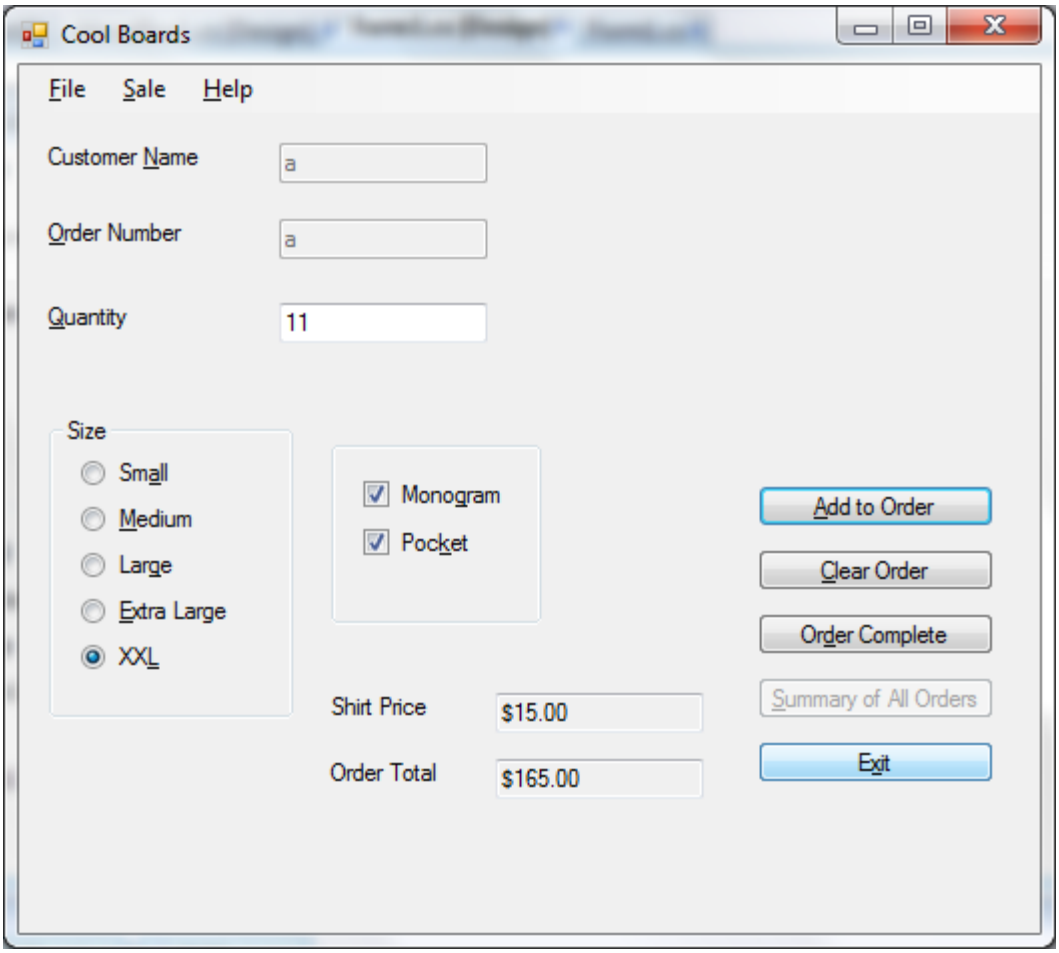
Shirts Sold: 40

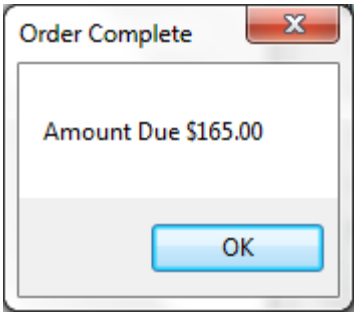
Number of Orders: 1

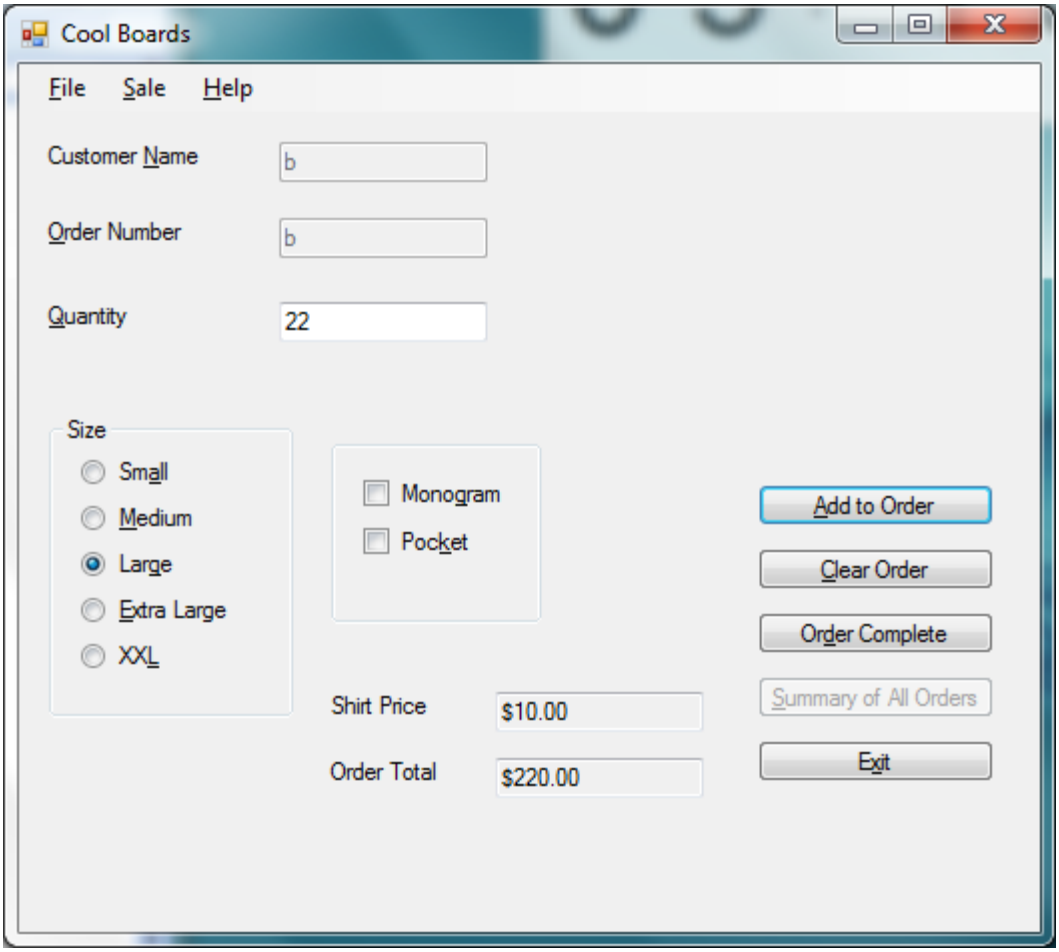
Total Sales: $400.00

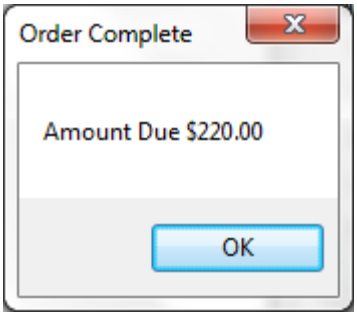
OK

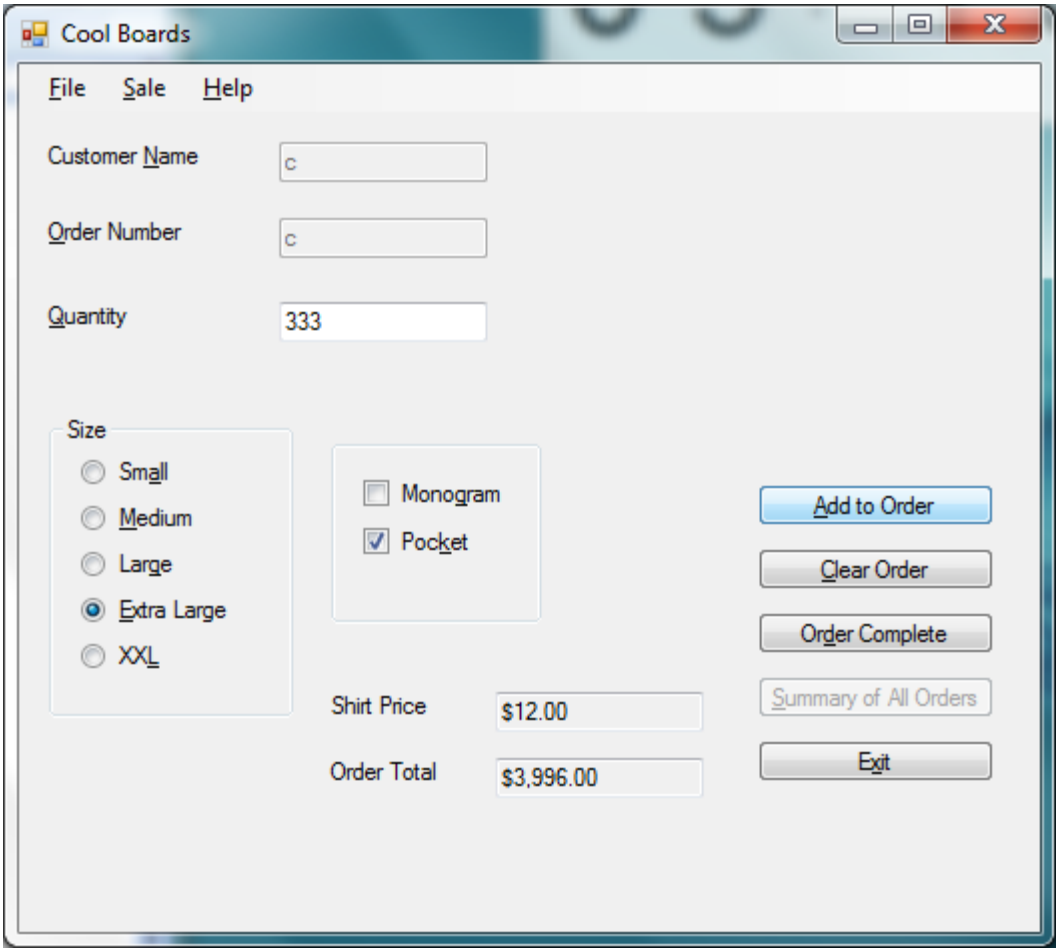
Test case #3
A orders 11 XXL shirts with monograms and pockets
B then orders 22 large shirts
C then orders 333 XL shirts with pockets.

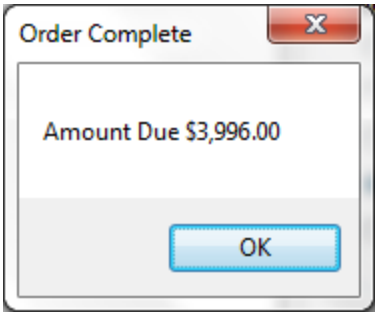


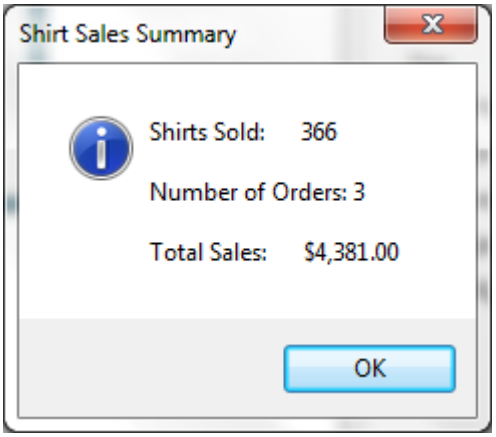



Cool Boards
File
Sale
Help
Customer Name
b
Order Number
b
Quantity
22
Size
Small
Medium
Large
Extra Large
XXL
Monogram
Pocket
Add to Order
Clear Order
Order Complete
Summary of All Orders
Exit
Shirt Price
$10.00
Order Total
$220.00

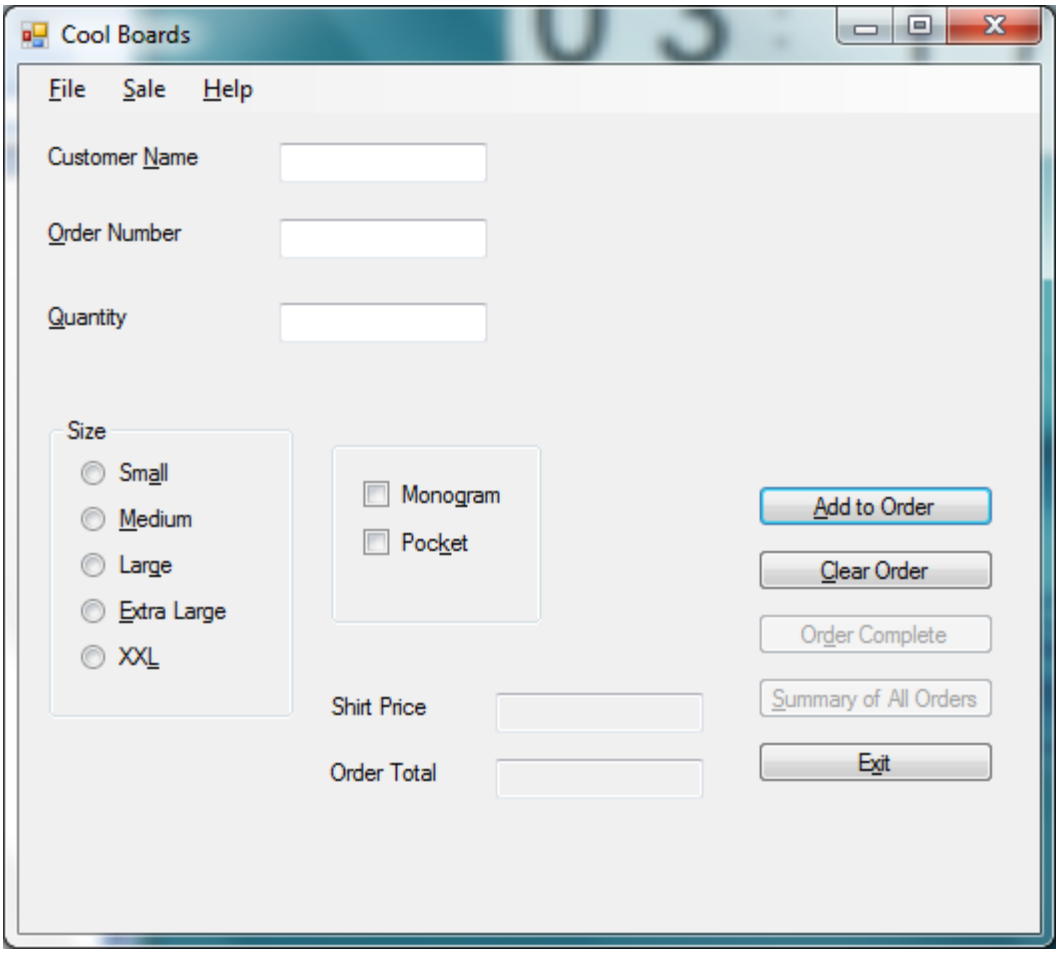

Order Complete
Amount Due $220.00
OK

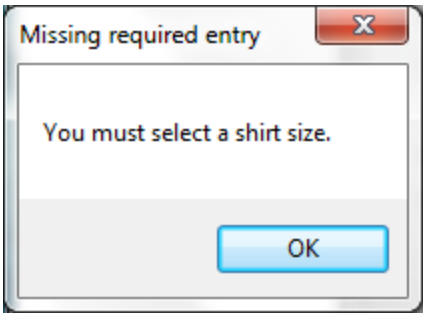

Cool Boards
File
Sale
Help
Customer Name
c
Order Number
c
Quantity
333
Size
Small
Medium
Large
Extra Large
XXL
Monogram
Pocket
Add to Order
Clear Order
Order Complete
Summary of All Orders
Exit
Shirt Price
$12.00
Order Total
$3,996.00

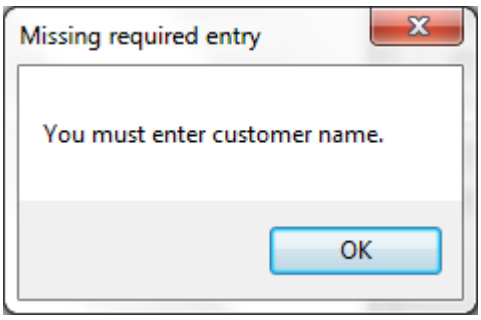

Order Complete
Amount Due $3,996.00
OK


Shirt Sales Summary
Shirts Sold: 366
Number of Orders: 3
Total Sales: $4,381.00
OK

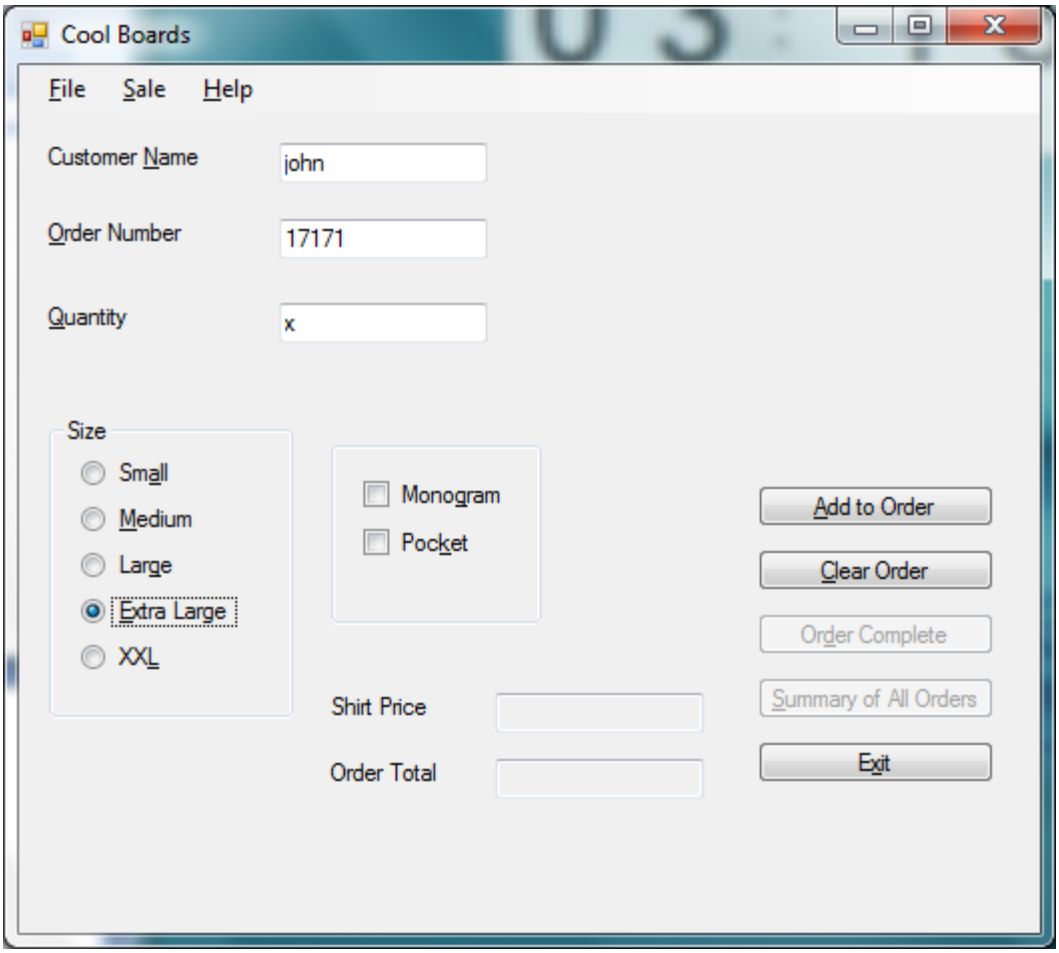
Test case #4
Person forgets to enter data and just clicks add to order

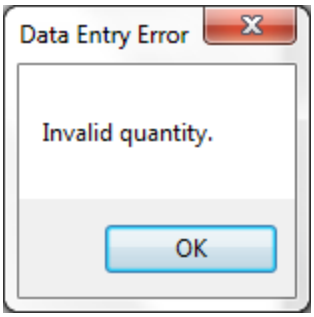






Test case #5
John writes x as a quantity





CISP 41 Programming in C#

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 = "";
            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",
```

```
            MessageBoxButtons.YesNo,
            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();
}
        }
}
```