Fall 2012
Points Possible: 100

Directions:
You will have 3 hours to complete this exam.

- From Question 1 download and save the questions (pdf) **and** the zip file.
- Extract (unzip) the zip file and open up the solution, and run it. On your monitor you will see Problem 1, Problem 2, etc. This is just a skeleton template that I have provided.
- Verify that the program works since you only have one attempt.
- Answer Question 1 by copying and pasting in your "Started" time as shown when you started the quiz.
- Save your answer and submit the quiz.
- When you have finished answering the questions, or have run out of time, submit your source file (yourLastNameQuiz2.cs) to the assignment named “Quiz 2 Submission” [emails were too confusing, so I’m using the Assignment tool].
- The time stamp of when you started the quiz and the time stamp of your assignment submission will determine the length of time.
- For each minute late, I will deduct one point from the quiz, so please manage your time, and give yourself extra time to submit your quiz to me (within the three hour period)

For this assessment, you will be modifying an existing project. This project is an open book assessment. That is to say you may use your book and notes for this course. You may NOT, however, correspond/talk to anyone about anything during the course of development and during the open time period. You may not receive answers or help from anyone. You must do your own work. You may email me for any specific question, but unfortunately, I may not respond in time for you to complete the assessment.

You will have 3 hours to work on the project. Make sure to set aside that amount of time to complete your work and manage your time. After retrieving the directions, the zip file, and "answering" the question with the Started time, click the "Submit" button to "finish" your exam.

Remember that the Software Development Standards still apply in regards to variable naming conventions, indentation, and use of decision structures. The only exception is that internal comments are optional.

Submit the source file only as an attachment to the specified assignment.

Fall 2012
Points Possible: 100

Add comments at the beginning of the code to include today's date and your name. For this assessment, use the **TryParse** method to convert data input. For all data displayed on the screen, use **placeholders**. For numeric data, format using the appropriate character **format** within the placeholder. Use the exact prompts as shown for each problem. You are not allowed to use class level variables, use only local variables within the curly braces.

Change the name of the source file to *yourLastName*Quiz2.cs (e.g. UnwinQuiz2.cs)
Leave the class name as Quiz2
Note: "real numbers" are numbers that may show fractional values (decimal places)

Problem 1 (30 points) – no methods required
When the user runs this block of code, the values in the array will be displayed, along with two numbers, a sum value, the lowest value, and the number of occurrences of the highest value. Appropriate data types are required, and use the Length property of the array to control all looping constructs except the 'foreach'. Here are the steps to follow:

a. Declare and instantiate an **integer** array of size 25 with the name **arrNum.** Declare an integer constant for the SIZE of the array. Make sure to use the "new" operator.

b. Use a 'for' loop to populate the array with random numbers from 1 to 10.

c. Use a 'do' loop structure to accumulate the values that are in the array. Store that value in a variable named **sum**.

d. Do not sort the array, and use a 'while' loop to determine the lowest value in the array. Keep it simple, you do not have to create a sorting routine.

e. Use a foreach loop to display the values in the array on one line with spaces between each number. (See below) Within this loop, count the number of times the lowest value occurs in the array.

f. Display the sum, the lowest value and the number of occurrences of the lowest value. (See below)

Required output [values listed will vary]:

```
 Problem 1

5   3   1   8   5   1  10   1  10   2   6   1   4   6   4   2   4   6
8   1   7   6   2   1   1
 Sum of values: 105
 Lowest value: 1 occurs 7 time(s).
```

Problem 2 (40 points)
Description: Use two single dimension arrays, where one will be a Lookup Table to determine the cost of an individual ticket in a specified seating section. The user will input the section (string) and the number of seats (integer) ordered; then the program will determine the price

(real number) for the section and display the total cost (real number) for the tickets. This will use methods, and you do not need to validate the data. Define the methods for this question directly under Main and above the Pause() method, and in the same Class. Use the following pseudocode to guide you.

a. Declare and initialize two arrays, arrSection (Section) and arrTicketPrice (Ticket Price) and use the "new" operator when declaring the arrays. Use the following data to initialize the arrays and do not change the order as shown.

Section	Ticket Price
orchestra	60.50
mezzanine	41.75
general	35.00
balcony	25.25

Use 1 C# statement for each array to declare and initialize the values in them. Declare the arrays within Main()

b. Method **GetSection –** Parameters: none; return: seating section.
Prompt for the seating section.

c. Method **GetNumberOfTickets -** Parameters: none; return: number of tickets
Prompt for the number of tickets.

d. In Main(), use the arrSection array to determine the subscript/index value to use for the ticket price. Use a 'while' repetition structure and not the binary search to determine the index value to use as a Lookup in the ticket price array.

e. If the user inputs an invalid section, display an error message that echos the value the user entered. See example after this question. Do not add a loop to ensure valid data.

f. If the data is valid, call the **CalcTicketCost** method (see step h).

g. Display the total cost formatted with a $ sign and two decimal places.

h. Method **CalcTicketCost** - Parameters: the number of minutes, the cost per minute; returns: the calculated total cost. There is a "process fee" included in the total cost of the ticketsl of $1.50 (which will not change). This method will not get nor display any data to the console. Create and use a constant for the process fee.

See examples of the prompts, and text below. Use Write instead of WriteLine for the prompts.

Required display :

```
     Problem 2
Enter the seating section: orchestra
Enter the number of tickets: 6

Total cost for the tickets are: $364.50
```

Required display [Error]:

```
      Problem 2
Enter the seating section: orchestration
Enter the number of tickets: 10

Invalid entry, orchestration section does not exist.
```

Problem 3 (30 points)

Description: Within Main, Problem 3 curly braces, this section will call 2 methods which will be created directly below CalcTicketCost() method and right above the Pause() method. Do not declare any class level variables; use only local variables in each code block and method.

a. In this code block, prompt the user for three integer values and store into appropriate data type variables. You do not need to validate the data, but you must use TryParse, and numbers can be positive or negative. Use Write instead of WriteLine for the prompts.

Within the class, create the 2 methods as describe: Product(), Division()

b. **Product()** with three parameters. The parameter names must be different than the variables stored in the Problem 3 code block.
Calculate the product and return the value

c. **Division()** with two parameters – the first parameter is the dividend, the second parameter is the divisor. The parameter names must be different than the variables stored in the Problem 2 code block.
In this example (200/7), 200 is the dividend, and 7 is the divisor
Calculate the result (no truncated values) and return the value.
Check for zero division. To keep this simple, if this error occurs, return a negative 999.99.

d. In this Problem 3 code block, call each of the methods, with the following arguments:
    a. Product: use all three input numbers as arguments
    b. Division: use the "last two" input numbers as arguments
e. Use one WriteLine to display both resulting values with appropriate labels. Show the product with commas, and no decimal places, except show the division result with 2 decimal places.

Required display [example1]:

```
      Problem 3
Enter number 1: 77
Enter number 2: 55
Enter number 3: 0

The results of the two method calls are:
      Product: 0
      Division: -999.99
```

Required display [example2]:

```
      Problem 3
Enter number 1: 77
Enter number 2: 55
Enter number 3: 33

The results of the two method calls are:
      Product: 139,755
      Division: 1.67
```

Challenge (10 points)

This problem uses a single conditional ternary operator. (Solutions that do not use this operator will receive no credit for this challenge.)

a. Prompt the user for their savings account balance. The user will enter valid numeric digits (positive only, real numbers)
b. If a person has saved more than $10,000, then their interest rate is calculated at 4% of their balance. If the person has saved $10,000 or less, then their interest rate is calculated at 2% of their balance.
c. Display the balance, and the interest rate amount as a number, commas, and 2 decimal places (no dollar sign). [see an example below]
d. Use a constant for the threshold of $10,000.
e. The user will enter valid numerical digits (positive only). Use of variables and appropriate data types are required. No data validation is required, though you must sill convert the data to numbers. To receive credit, do not embed the ternary conditional operator inside the CW.

Required display:

```
Enter your savings account balance: 12456.12
Interest of 493.80 was added to your balance of 12,456.12
```

Submission

***** do not zip the solution folder; do not zip anything ***

1. Go to the **Assignment** called “Quiz 2 Submission” in Canvas, and submit the source file only [yourLastNameQuiz2.cs] as an attachment.

It is your responsibility to ensure that the correct file is submitted. There is no time for me to check, validate and request a new file from you. If I don’t have the correct file, I cannot grade your project and therefore, you will receive a zero for this quiz. There is a penalty of one point deducted for each minute late.