Introduction to C# Midterm

Winter 2013

Name: _____

Score: ____ / 100

Section 1	Section 2	Section 3	Extra Credit

Section 1: General Concepts (40 pts)

Answer the following questions by typing or writing a short paragraph. Be concise.

1. What's the difference between a class and an interface?

2. Consider the following declaration of an instance field on an object:

   private readonly int age;

   What is the significance of the "readonly" modifier?

3. What's the difference between a static method and an instance method? If you were given the following class definition, write code to call each of the SayHello* methods.

```
public class Greeter
{
   public static void SayHelloStatic() { Console.WriteLine("Hello static");  }
   public void SayHelloInstance() { Console.WriteLine("Hello instance");  }
}
```

4. Given the following declarations describe what's wrong with the StackImplementation class.

```
public interface IStack<T>
{
   int Count { get; }
   void Push(T toPush);
   T Pop();
}

public StackImplementation<T> : IStack<T>
{
   public void Push(T toPush) { // Implementation details omitted for brevity }
   public T Pop() { // Implementation details omitted for brevity }
}
```

5. What is a generic type?

6. List and describe the scope of the four access modifiers for Types, Methods, and Fields.

7. What's the difference between a for loop and a foreach loop?

8. What's the difference between a while loop and a do/while loop?

9. When writing code to work with collections of things can either use an array of objects (object[]) or an IList<T> where T is the type you wish to hold in your list. What's the advantages of the second approach over the first?

10. What is a nullable type?

Section 2: Syntax & Class Design (30 pts)

Consider the following class and interface declarations. Answer the following questions about them. Note that pieces of this class hierarchy have been left blank where implementations should be. That is so that the example code fits on the page.

```
public interface IDrawable
{
   void Draw(Canvas toDraw);
}

public abstract class Shape : IDrawable
{
   // Constructor
   public Shape(string id) { // Implementation goes here }

   // Properties
   public string Id { get; private set; }
   public abstract double Area { get; }

   // Methods
   public abstract void Draw(Canvas toDraw);
}

public class Triangle : Shape
{
   private readonly double area;

   public Triangle (string id)
      : base(id) { // Implementation goes here }

   public double Area{ get { // Implementation goes here }}

   public void Draw(Canvas c) { // Implementation goes here }
}

public class Canvas : IDrawable
{
   private IList<Shape> myShapes = new List<Shape>();

   public static Shape CreateShape(string id) { // Implementation goes here }
   public void Draw(Canvas toDraw) { // Implementation goes here }
}
```

1. What is the name of the interface that the Shape class implements?

2. What does “abstract” mean on the Shape class type declaration? What about on the Draw method declaration?

3. What does ***: Shape*** to the right of the Triangle class Type definition mean?

4. What’s the difference between the Id property on the Shape class and the Area property on the Triangle class?

5. Is the following definition legal? Why or why not?

   ```
   Triangle myTriangle = new Triangle(“123”);
   string myId = myTriangle.Id;
   ```

6. What is a field initializer? In which class above (Shape, Triangle, or Canvas) do we use a field initializer to initialize an instance field?

7. What is a constructor? Given the class hierarchy above, if we were to instantiate a new Triangle class, in what order would the class constructors be called? *Hint: You are given the fact that both the Shape and the Triangle constructors are called you just need to dictate the order.*

Section 3: Collections & Data Structures (30 pts)

Answer the following questions about collections.

1. What's the difference between an Array and a List?

2. What are the advantages of using a Dictionary?

3. Given the list of elements below, in what order would they be returned if we were to push them onto a stack from left to right, and then pop them off and print? What about a Queue?

3	4	19	45	25

4. What is a linked list? Why are they useful?

5. Why can we use a foreach loop to iterate through the members of any collection in the entire .NET framework? *Hint: There's a top level interface that makes this magic possible, what is it?*

Extra Credit (20 pts)

Answer the following questions for extra credit. Note that they by design are more difficult than the other questions.

1. What is the ternary operator? Give an example.

2. What is the null coalescing operator? Give an example.

3. What is auto-boxing and auto-unboxing of primitive types? Give an example of when each occurs.

4. Explain the difference between reference types and value types.