Provide short answers to the following questions:

*Enumerations and Structures*

1. **What is the advantage of an enum over (for example) an int?**

An enum lets you create a value type that can be associated with a symbolic name. For example, {spring, summer, fall} rather than {0, 1, 2}.

1. **When might you prefer a struct over a class?**

You might prefer a struct over a class when you would prefer to use a value type over a reference type. Use structures for small data values for which it’s just as or nearly as efficient to copy the value as it would be to copy an address.

1. **What happens if you create a struct without using the new keyword?**

The struct will still be created but its fields are left in their uninitialized state, and any attempt to access the values in these fields will result in a compiler error.

1. **What is the purpose of CIL?**

The CIL is a platform-independent middle language for the platform-specific CLR. The CIL is a pseudo-machine code with a set of instructions converted by C#.

1. **What does the CLR do?**

The CLR is responsible for converting the CIL instructions into real machine code for the processor to use.

1. **What alternative to native code does the Universal Windows Platform provide for unmanaged applications?**

The alternative to native code that UWP provides for unmanaged apps is WinRT

**See next page for Questions 7-8**

**See jpeg for Output**

1. Define a class with:
   * One public field
   * One private field
   * A default constructor
   * A 2-argument constructor which sets your fields
   * A deconstructor

class Hw6d

{

public int fieldInt;

private string fieldString;

public Hw6d() // default constructor

{

Console.WriteLine("This is the default constructor for Homework 6D");

}

public Hw6d(int param1, string param2)

{

this.fieldInt = param1;

this.fieldString = param2;

}

public void Deconstruct(out int fieldInt, out string fieldString)

{

fieldInt = this.fieldInt;

fieldString = this.fieldString;

}

}

1. Instantiate an instance of your class using the 2-argument constructor. Deconstruct the instantiated object into a tuple

class Program

{

static void Main(string[] args)

{

var homework = new Hw6d(1, "Lee");

var (hwNum, hwString) = homework;

Console.WriteLine($"{hwNum}, {hwString}");

}

}