C# 3.5

**TIPS:**

**1. No ‘n+1’ and ‘n-1’ concept will work out. So please check the exact number of options they ask to choose.**

**2. Read Questions properly.**

1. Which statements are true about yield statements? (select 4)
2. A yield statement can also appear in an anonymous method.
3. A yield statement cannot appear in an anonymous method.
4. When used with expression, a yield return statement cannot appear in a catch block or in a try block that has one or more catch clauses.
5. A yield statement can only appear inside an iterator block.
6. The yield keyword signals to the compiler that the method in which it appears is an iterator block.

Answer: - b, c, d, e

1. Extension methods gives you the capability of adding methods of existing types of our own types, without creating the new derived class.

Answer: - True (These are the special type of static methods)

1. Query expressions can be compiled to expression trees or to delegates.

Answers: - True

1. A generic type parameter that is not marked covariant or contra variant is reference to as\_\_\_\_\_\_\_\_\_\_\_\_\_\_.
2. Nonvariant
3. Variant
4. Neutral
5. Invariant

Answer: - d (http://msdn.microsoft.com/en-us/library/dd799517.aspx)

1. Which statement is true about AssemblyBuilder class? (select 3)
2. The extended modules in the existing assembly are saved using the save method.
3. Define and represents a dynamic assembly.
4. The dynamic modules in the assembly are saved when the dynamic assembly is saved using save method.
5. Extends an existing assembly.
6. To generate an executable the SetEntryPoint method must be called to identify the method that is the entry point to the assembly.

Answer: - b, c, e

1. Generic keyword is used for covariance and contra variance.

Answer: - True

1. How to get the type of assembly?

Answer: - GetType()

1. How to define dynamic module of an assembly?

Answer: - DefineDynamicModule(string)

1. var keyword is use to initialize –

Answer: - Object

1. Which is the auto-increment able

(http://msdn.microsoft.com/en-us/library/system.collections(v=vs.71).aspx)

Answer: - ArrayList()

1. You can use the   \_\_\_\_\_\_\_\_ delegate to represent a method can be passed as a parameter, without explicitly declaring a custom delegate.

Answer: - Func(Of TResult) delegate

1. Which method gets serialization information with all of the data need to reinstantiate this assembly?

Answer: - GetObjectData

1. In a generic type definition the\_\_\_\_\_\_\_\_ clause is used to specify constraints on the type that can be used as arguments for a type parameter, defined in a generic declaration.

Answer: - Where (http://msdn.microsoft.com/en-us/library/bb384067.aspx)

1. List<int> digits= new List<int>{0,1,2,3,4,5,6,7,,8,9};

Answer:- Collection Intializer

1. Specify condition when partial classes become necessary

Answer: -

1. When working on large projects, spreading a class over separate files allows multiple programmers to work on it simultaneously.
2. When working with automatically generated source, code can be added to the class without having to recreate the source file.
3. To split a class definition, use the partial keyword modifier.
4. void Swap<T>(List<T> list1, List<T> list2)

{

//code to swap items

}

void Swap(List<int> list1, List<int> list2)

{

//code to swap items

}

Answer: - Open constructed and closed constructed types can be used as method parameters

1. for new() constraint :-

Answer:-

1. The type argument must have a public parameter less constructor.
2. When used in conjunction with other constraints, the new() constraint must be specified last.
3. Apply the new constraint to a type parameter when your generic class creates new instances of the type
4. List<int> numbers = new List<int>() { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };

var IEnumerable<int> filteringQuery =from num in numbers

where num < 3 || num > 7 select num;

Answer: -

a. Query expression demonstrates how to filter or restrict results by applying conditions with a where clause.

b. It returns all elements in the source sequence whose values are greater than 7 or less than 3.

1. String[] groupingQuery = { "carrots", "cabbage", "broccoli", "beans", "barley" }; IEnumerable<IGrouping<char, string>> queryFoodGroups = from item in groupingQuery group item by item[0];

Answer: - Expression demonstrates how to group results according to a key. This query returns two groups based on the first letter of the word.

1. Difference b/w action<T> and func<TResult>

Answer: - action<t> Encapsulates a method that has a single parameter and does not return a value.

Func(TResult: )Encapsulates a method that has no parameters and returns a value of

the type specified by the TResult parameter.

**NOTE:**

---🡪Main point to remember is that: Action does not return value and Func returns value.

-🡪it s Func(TResult) and not Function(TResult)

---🡪Example: IF Func(T,TResult):Has a single parameter(the parameter count is based on no of ‘T’ specified) and returns a value in TResult.

1. A flexible and secure method of isolating running applications.

Answer:- appdomain()

1. public static IEnumerable<TResult> Join<TOuter, TInner, TKey, TResult>(

this IEnumerable<TOuter> outer,

IEnumerable<TInner> inner,

Func<TOuter, TKey> outerKeySelector,

Func<TInner, TKey> innerKeySelector,

Func<TOuter, TInner, TResult> resultSelector

Answer: - Usage of Join operator

1. Generic and Non-Generic inherited from base class

Answer: - True

1. Interface which is type of covariant.

Answer:-

1. IEnumerable<T>
2. Iqueryable<T>
3. IEnumerator<T>
4. How non generic, generic is inherited from base class.

Answer: -

1. Generic classes can inherit from concrete, closed constructed, or open constructed base classes
2. Non-generic —can inherit from concrete, closed constructed base classes, but not from open constructed classes
3. What are true about query expression?

Answer:-

1. Query expressions are often more readable than equivalent expressions written in
2. Some query operations, such as [Count](http://msdn.microsoft.com/en-us/library/system.linq.enumerable.count.aspx) or [Max](http://msdn.microsoft.com/en-us/library/system.linq.enumerable.max.aspx), have no equivalent query expression clause and must therefore be expressed as a method call
3. Query expressions can be compiled to expression trees or to delegates
4. True about anonymous method.

Answer:-

Some option with: -- Use jump,goto ,break it is compiler error or not.

28. Which are not true about anonymous type?

The true facts are:-

1: Anonymous types contain one or more public read-only properties

2: Anonymous types typically are used in the [select](http://msdn.microsoft.com/en-us/library/bb384087.aspx) clause of a query expression to return a subset of the properties from each object in the source sequence. For more information about queries

\*\*\*\*\*\**You need to check option other than these two.*

29. Expression <Del> myet =X => X\*X represents

Answer:-

1. Expression tree.
2. Delegate type.

30. How to Creates an instance of the specified type 'whose name is specified' using the constructor

Answer:- CreateInstance()

31.var V item =new (Rate =90, msg="hello");

Answer: -

1.Example of anonymous type

2. anonymous type that is initialized with two properties named Rate and msg.

32. \_\_\_\_\_\_\_\_\_\_\_\_ Method locates the specified type from this assembly & creates instance using system.

Answer: - Create Instance()

33. True about Lambda Expression

Answer: -

1. A *lambda expression* is an anonymous function that can contain expressions and statements, and can be used to create delegates or expression tree types
2. The left side of the lambda operator specifies the input parameters (if any) and the right side holds the expression or statement block
3. And 1 more option …dnt remember … something like => operator .

34.Question on anonymous type:

Answer:-

1. If two or more anonymous types in the same assembly have the same number and type of properties, in the same order, the compiler treats them as the same type
2. They share the same compiler-generated type information.

35. Significance of dynamic method

Answer:-

1. Discarded methods are available for garbage collection.
2. It can define dynamic method that compile, executed and discarded.
3. use the DynamicMethod class to generate and execute a method at run time, without having to generate a dynamic assembly and a dynamic type to contain the method

36. On nested types

public class Container

{

public class Nested

{

private Container parent;

public Nested()

{

}

public Nested(Container parent)

{

this.parent = parent;

}

}

}

Answer:- Ans is within this (read it from msdn type the que):

A nested type has access to all of the members that are accessible to its containing type. It can access private and protected members of the containing type, including any inherited protected members.

the full name of class Nested is Container.Nested. This is the name used to create a new instance of the nested class, as follows

The nested, or inner type can access the containing, or outer type. To access the containing type, pass it as a constructor to the nested type

A type defined within a [class](http://msdn.microsoft.com/en-us/library/0b0thckt.aspx) or [struct](http://msdn.microsoft.com/en-us/library/ah19swz4.aspx) is called a nested type

Regardless of whether the outer type is a class or a struct, nested types default to [private](http://msdn.microsoft.com/en-us/library/st6sy9xe.aspx), but can be made [public](http://msdn.microsoft.com/en-us/library/yzh058ae.aspx), protected internal, [protected](http://msdn.microsoft.com/en-us/library/bcd5672a.aspx), [internal](http://msdn.microsoft.com/en-us/library/7c5ka91b.aspx), or [private](http://msdn.microsoft.com/en-us/library/st6sy9xe.aspx). In the previous example, Nested is inaccessible to external types, but can be made public like this:

The nested, or inner type can access the containing, or outer type. To access the containing type, pass it as a constructor to the nested type

Nested types can access private and protected members of the containing type, including any inherited private or protected members.

37.Var keyword is used for

Answer:-

* var can only be used when a local variable is declared and initialized in the same statement; the variable cannot be initialized to null, or to a method group or an anonymous function.
* var cannot be used on fields at class scope.
* Variables declared by using var cannot be used in the initialization expression. In other words, this expression is legal: int i = (i = 20); but this expression produces a compile-time error: var i = (i = 20);
* Multiple implicitly-typed variables cannot be initialized in the same statement.
* If a type named var is in scope, then the var keyword will resolve to that type name and will not be treated as part of an implicitly typed local variable declaration.

You may find that var can also be useful with query expressions in which the exact constructed type of the query variable is difficult to determine. This can occur with grouping and ordering operations.

The var keyword can also be useful when the specific type of the variable is tedious to type on the keyboard, or is obvious, or does not add to the readability of the code. One example where var is helpful in this manner is with nested generic types such as those used with group operations. In the following query, the type of the query variable is IEnumerable<IGrouping<string, Student>>. As long as you and others who must maintain your code understand this, there is no problem with using implicit typing for convenience and brevity

38. LINQ to DataSet

Given some query.So choose two options involving the keyword dataset

39..LINQ to XML

Given some query with some ‘load’ method.Choose option involving key words,

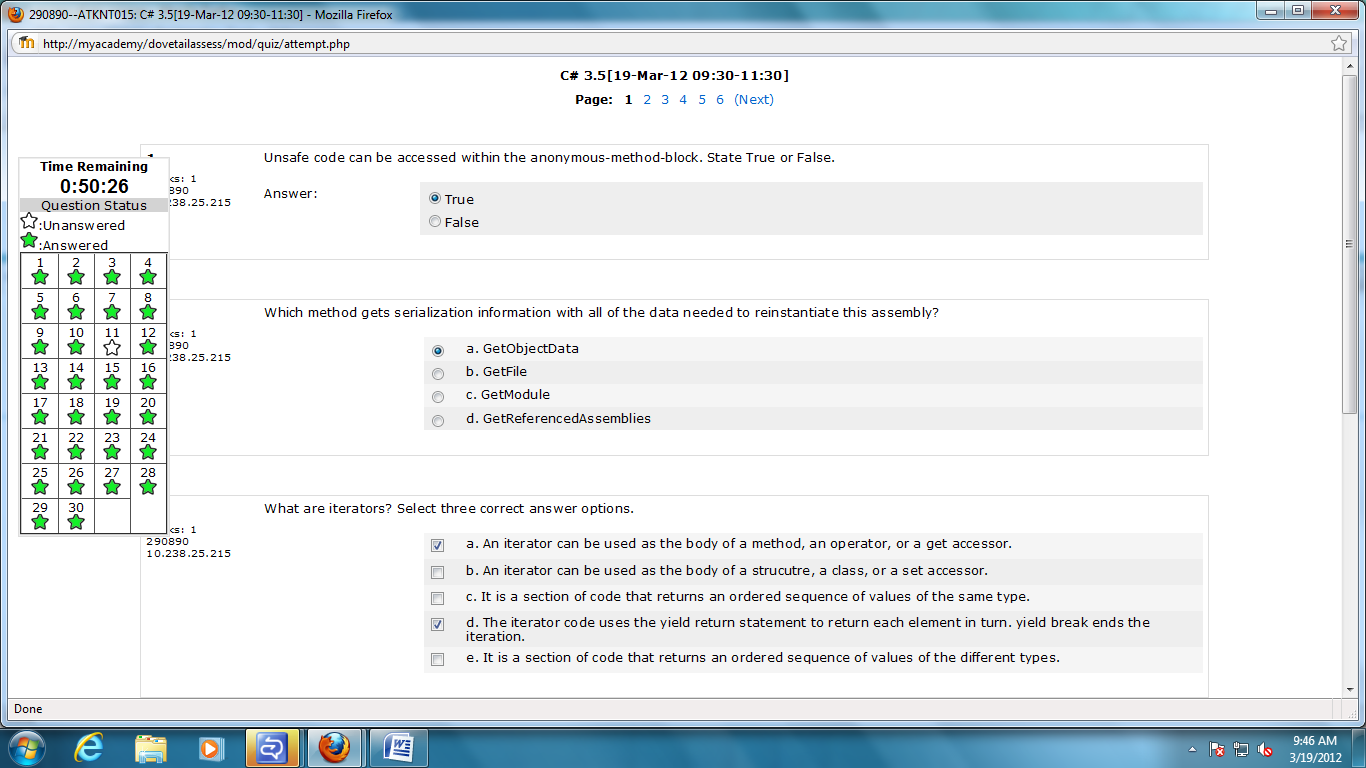
1.linq to xml and

2.element containing attribute value.

40. The preference for generic classes is to use generic interfaces ,such asTComparable<T>rather than IComparator,why??

To avoid boxing and unboxing operations on value type.

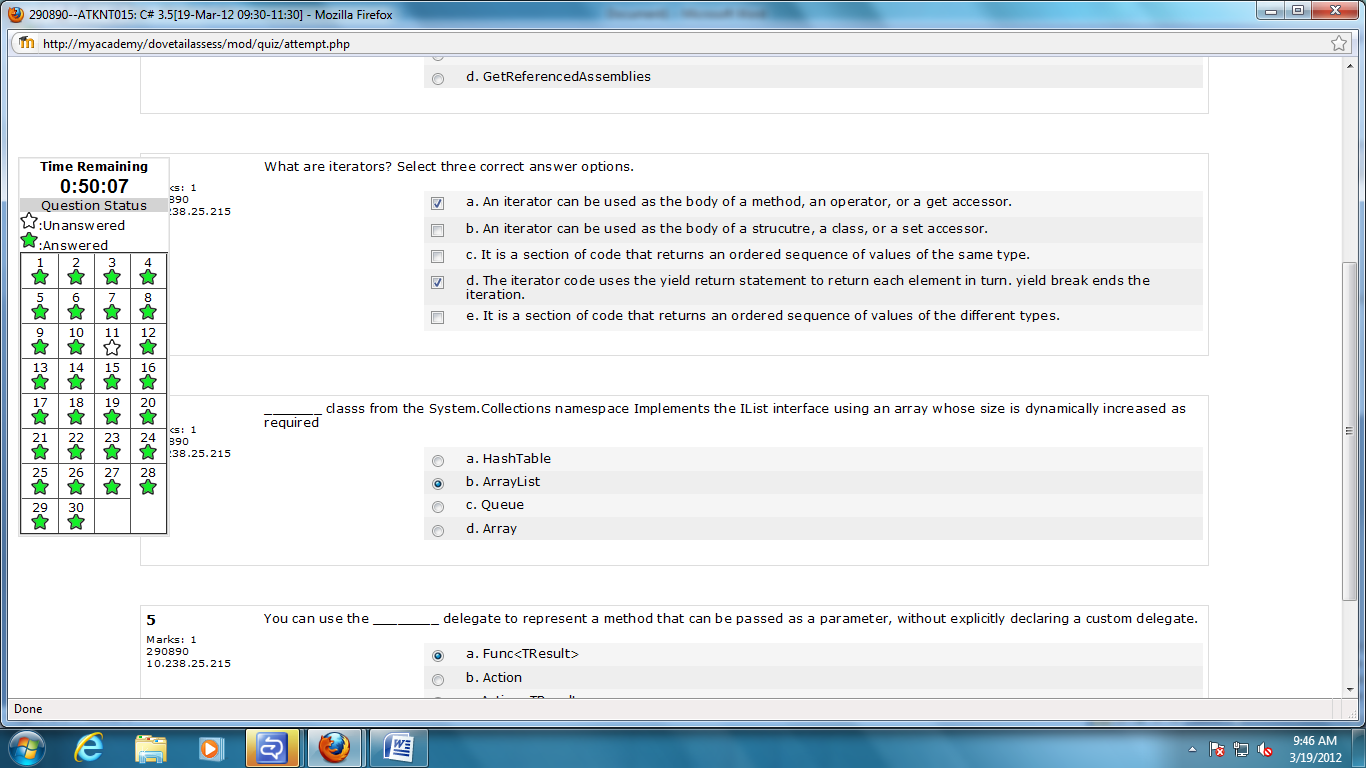
**ALL THE BEST☺**



1.false

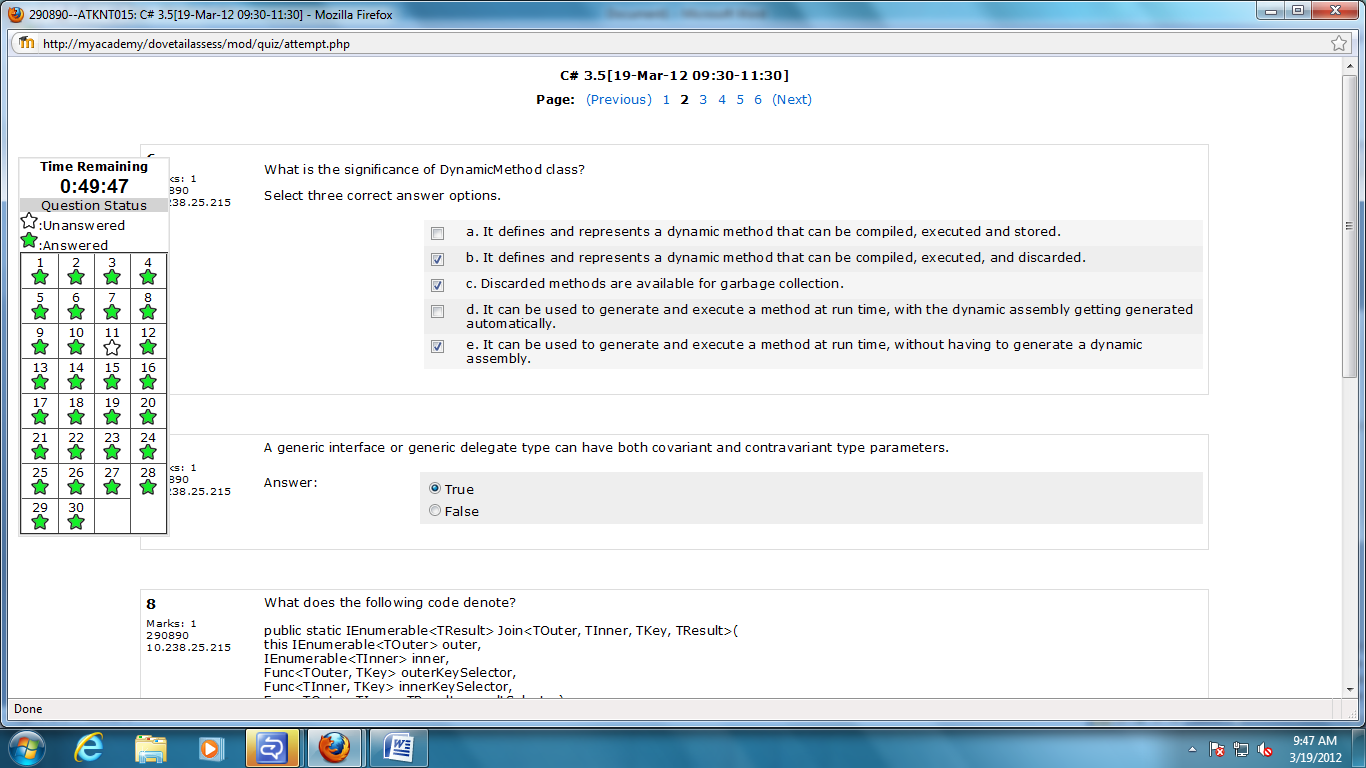
2.A

3.a,c,d



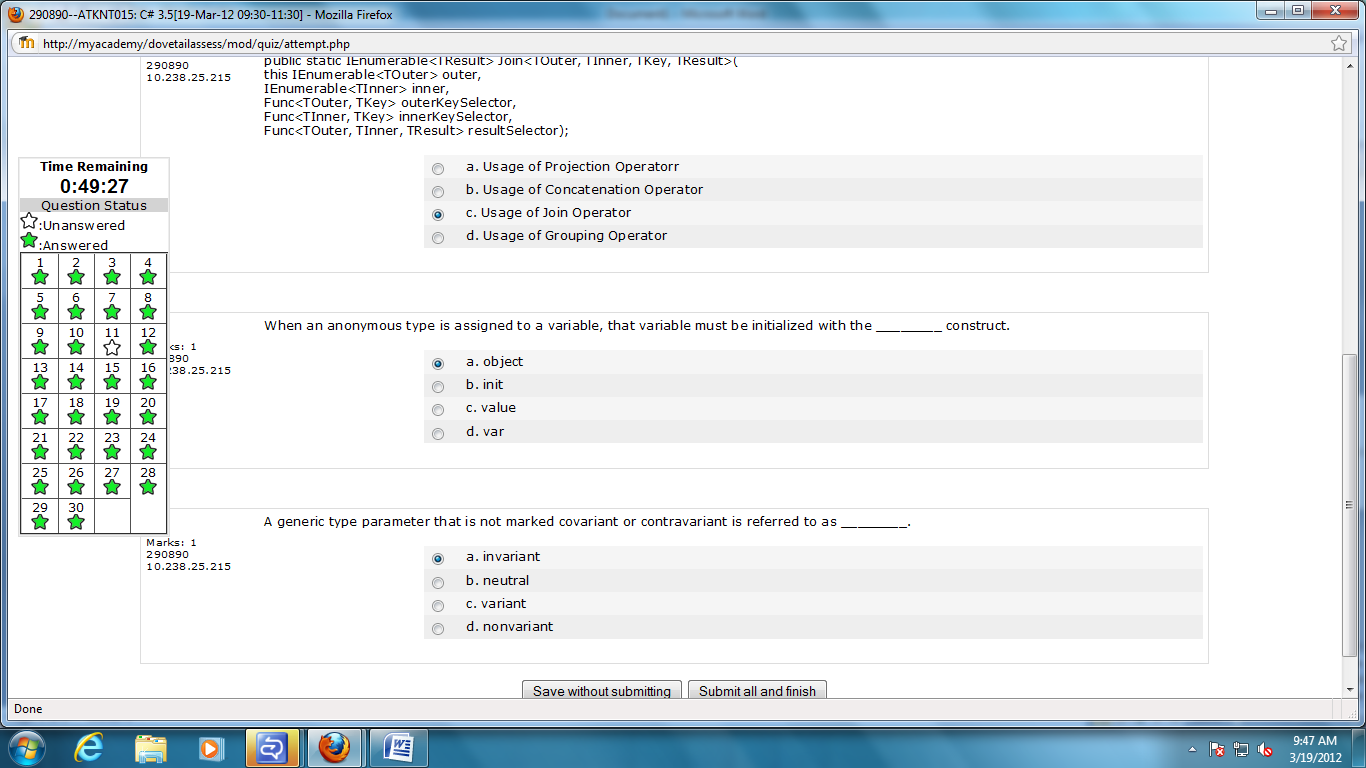
4.ArrayList

5.a.Func<TResult>



6.b,c,e

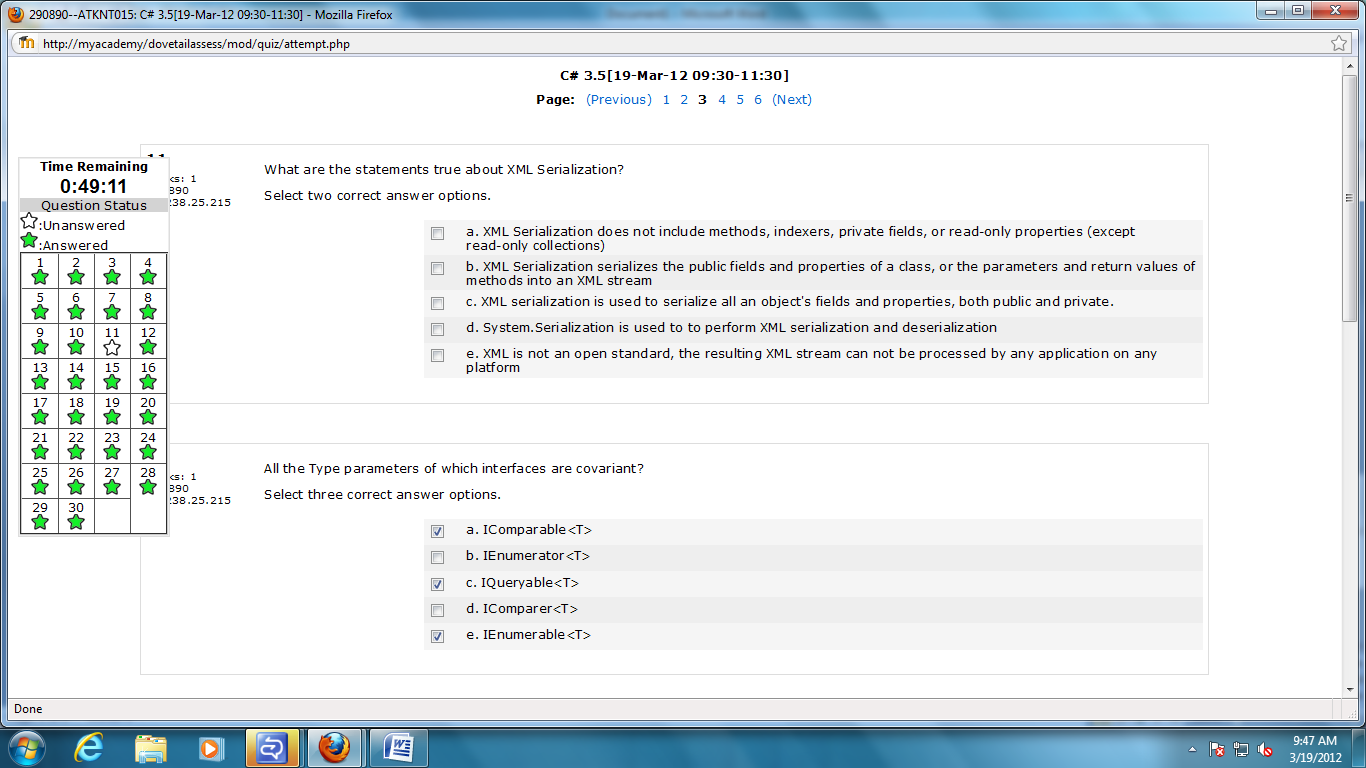
7.True



8.Usage of join operator

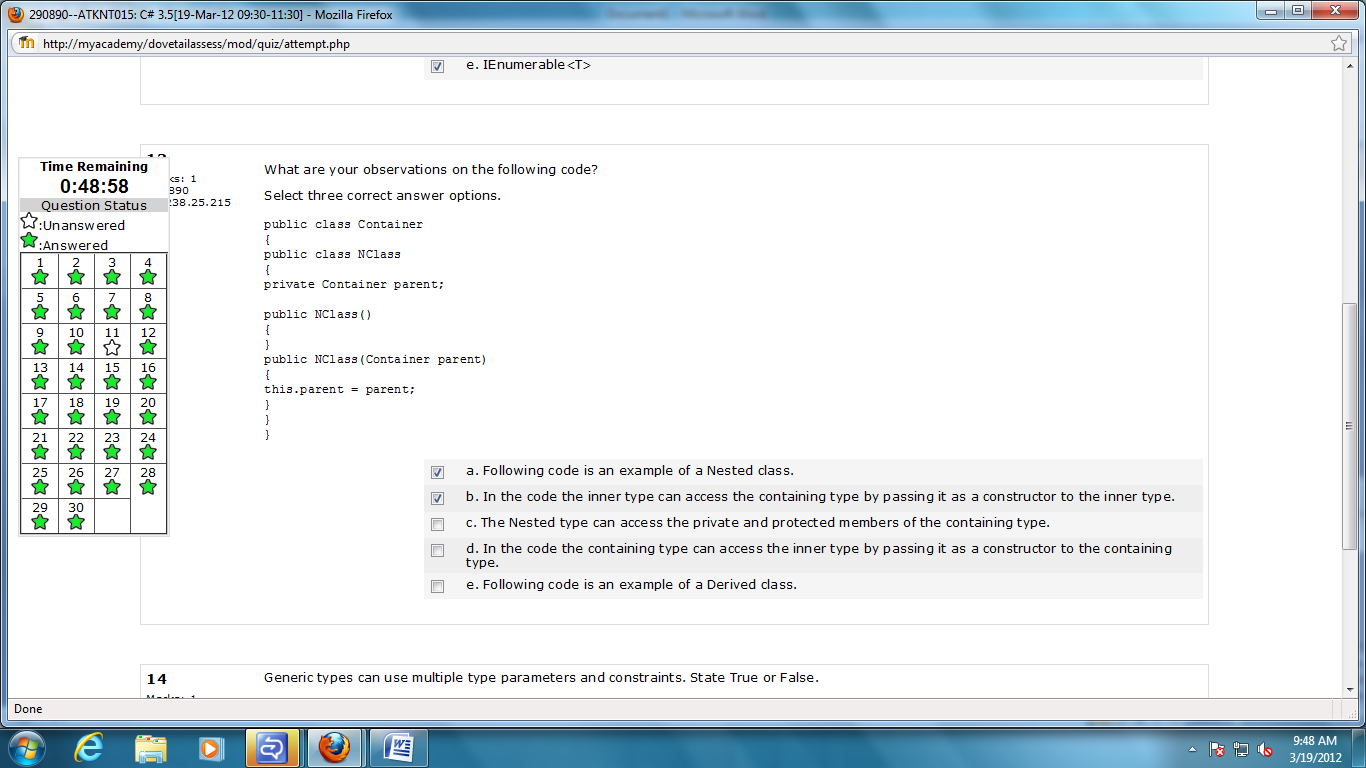
9.var

10.Invariant

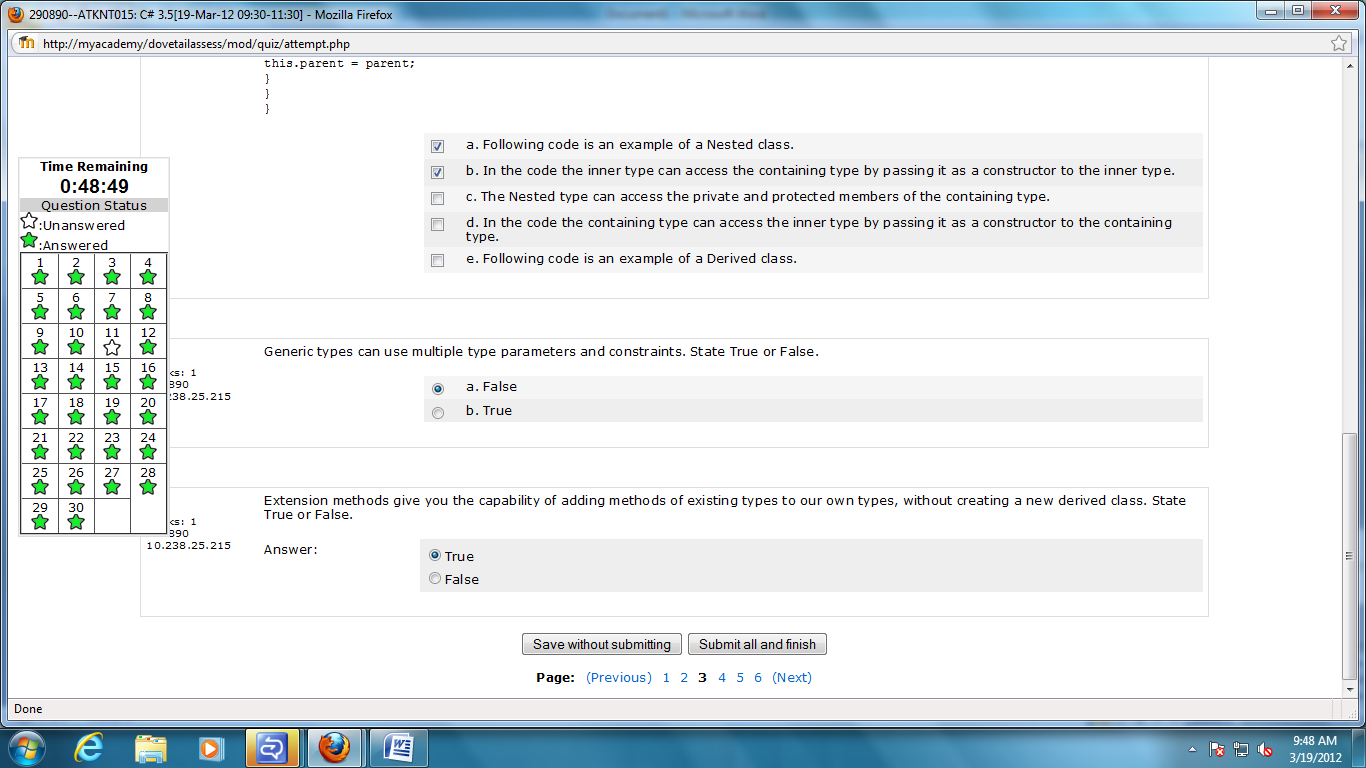


11.a,b

12 b,c,e

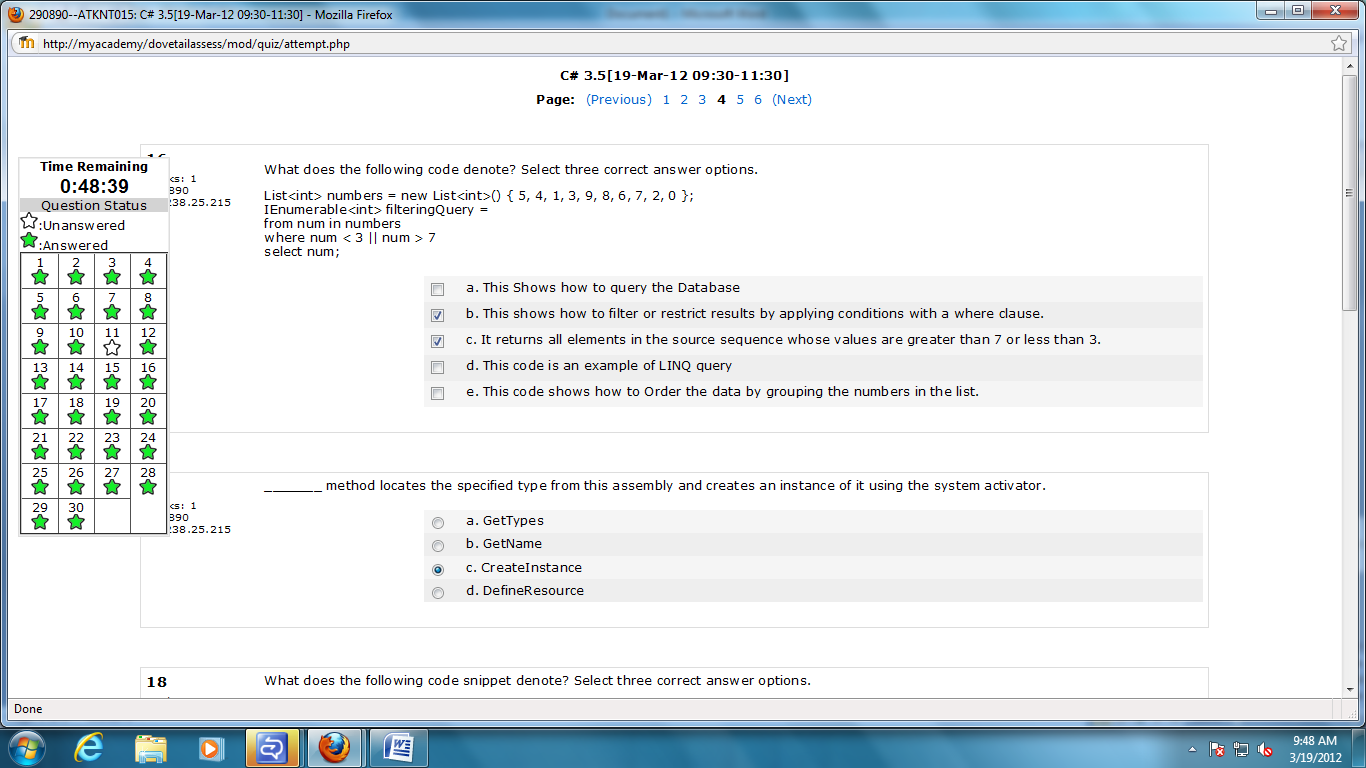


13 a,b,c



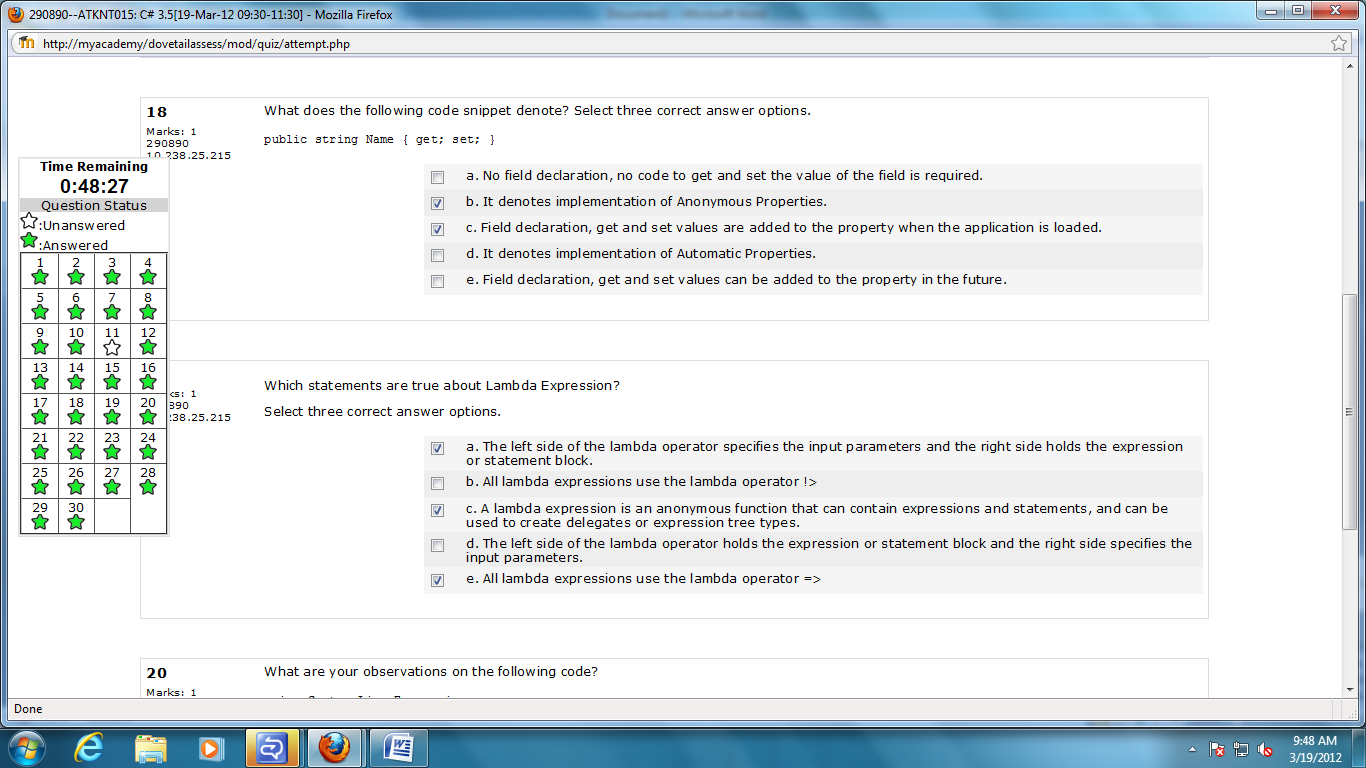
14.true

15.true



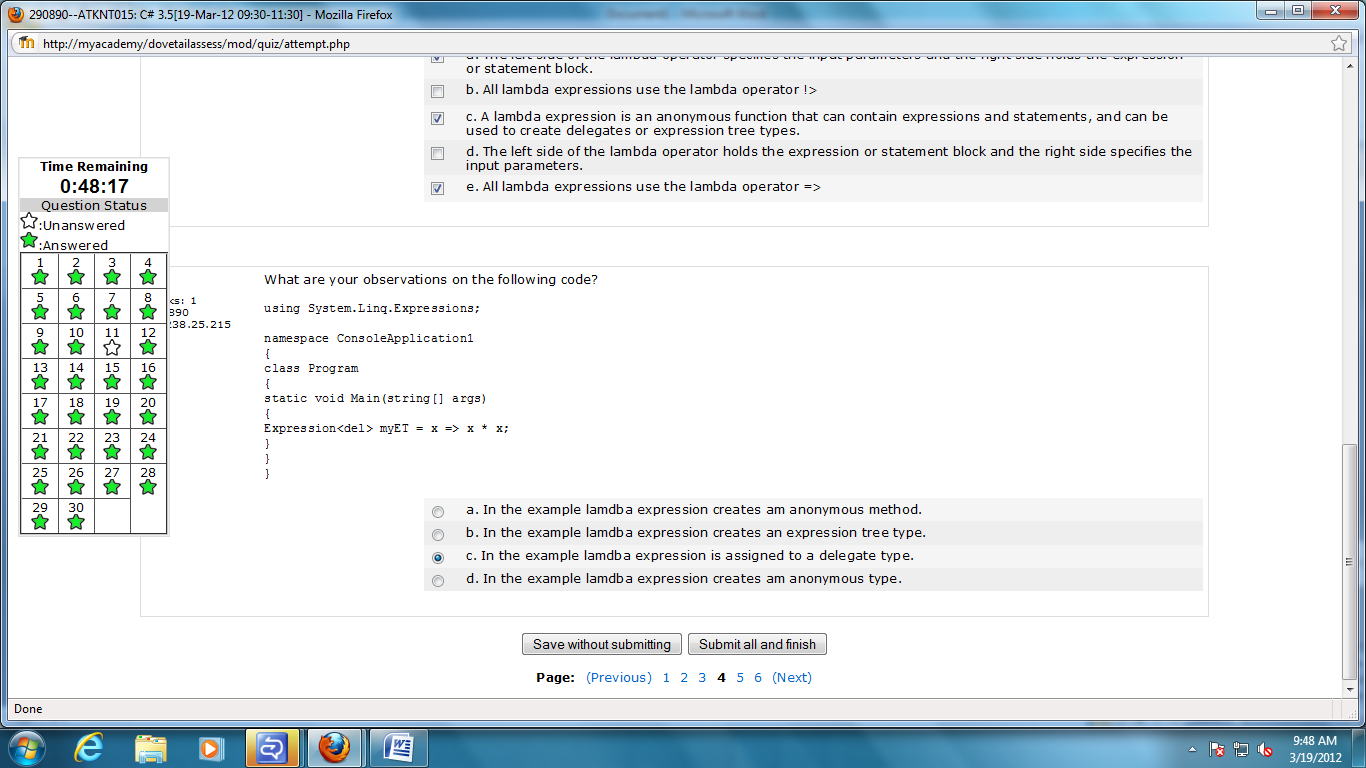
16.b,c,d

17.CreateInstance

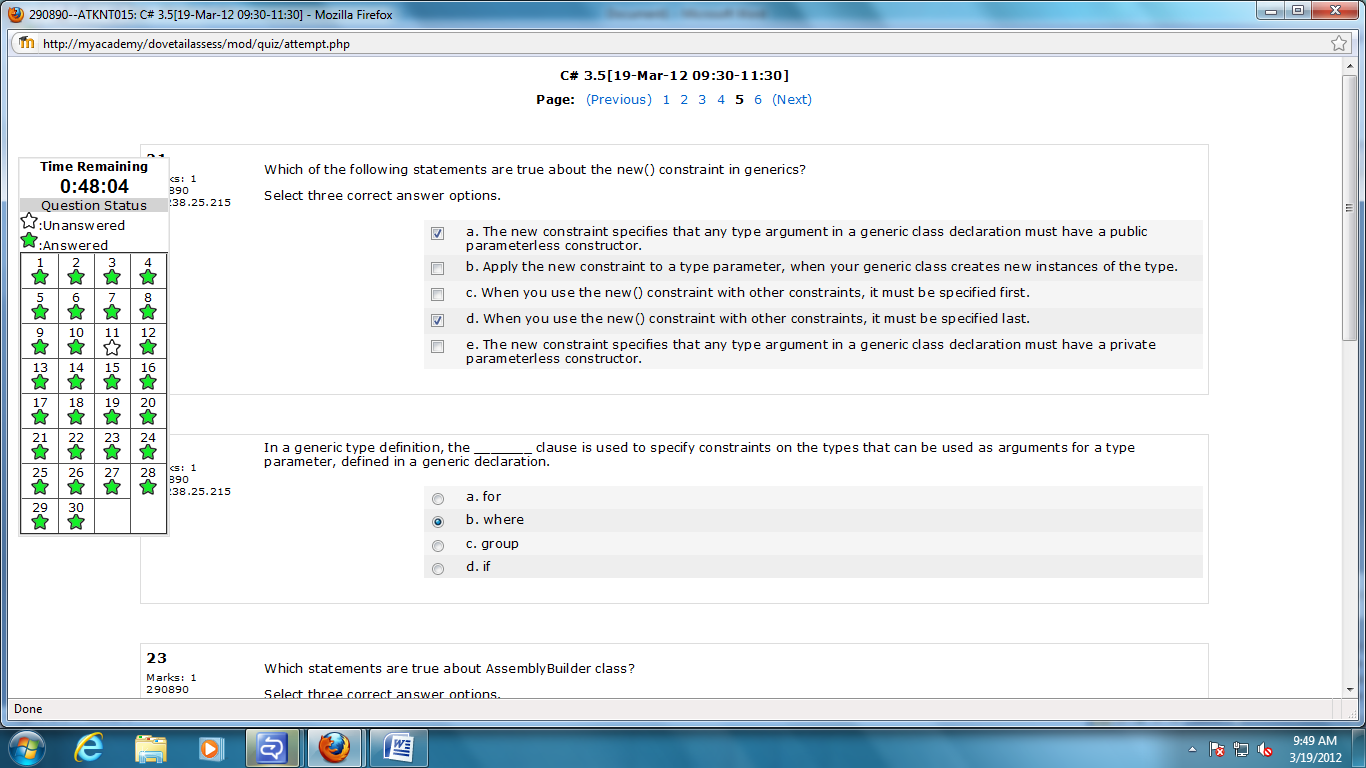


18.a,b,c

19.a,c,e

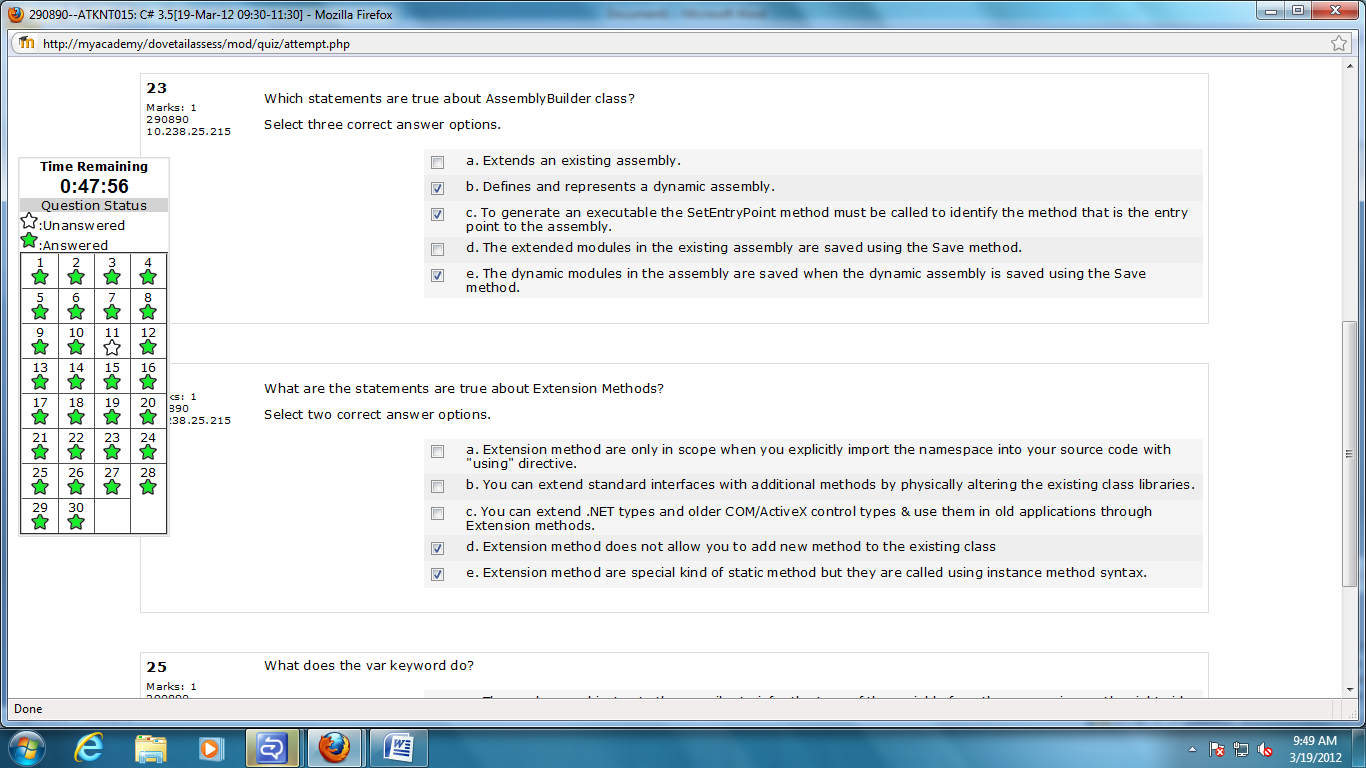


20.b



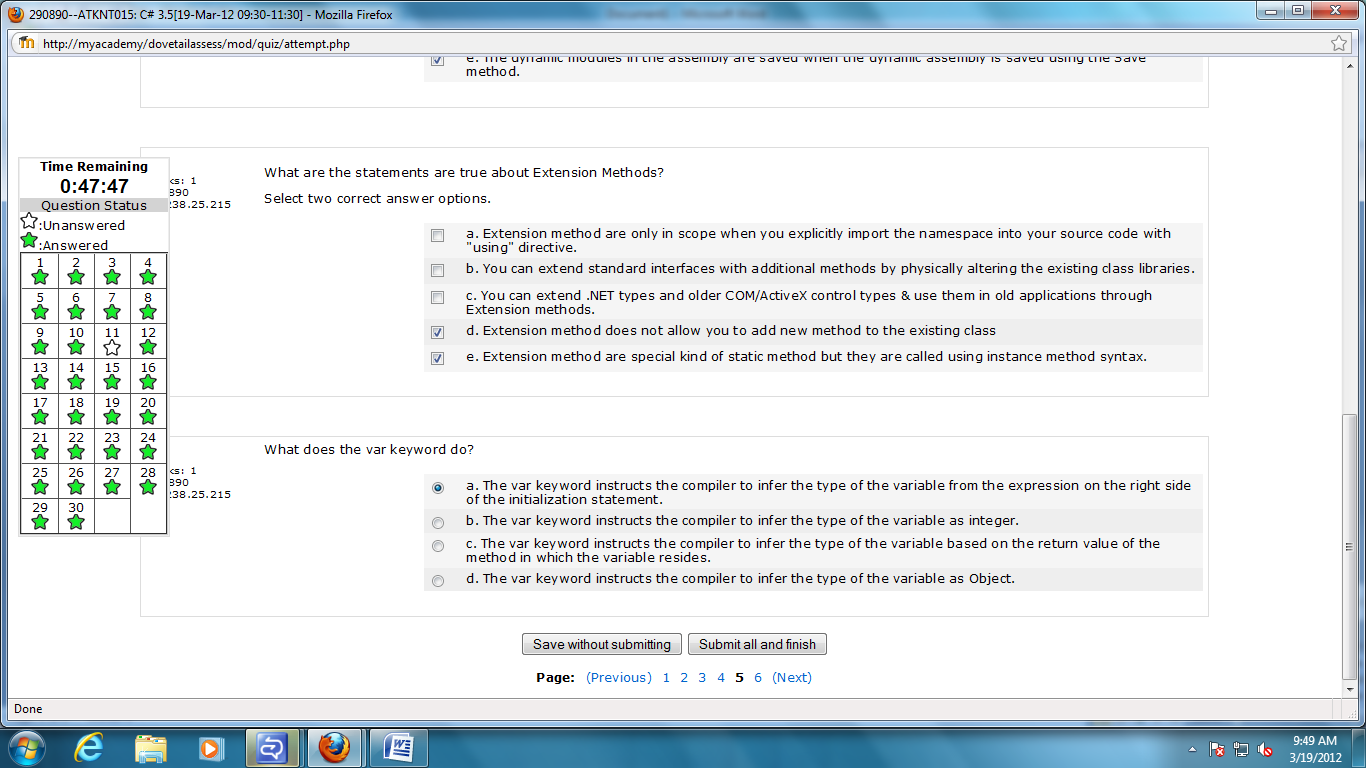
21.a,b,d

22.b

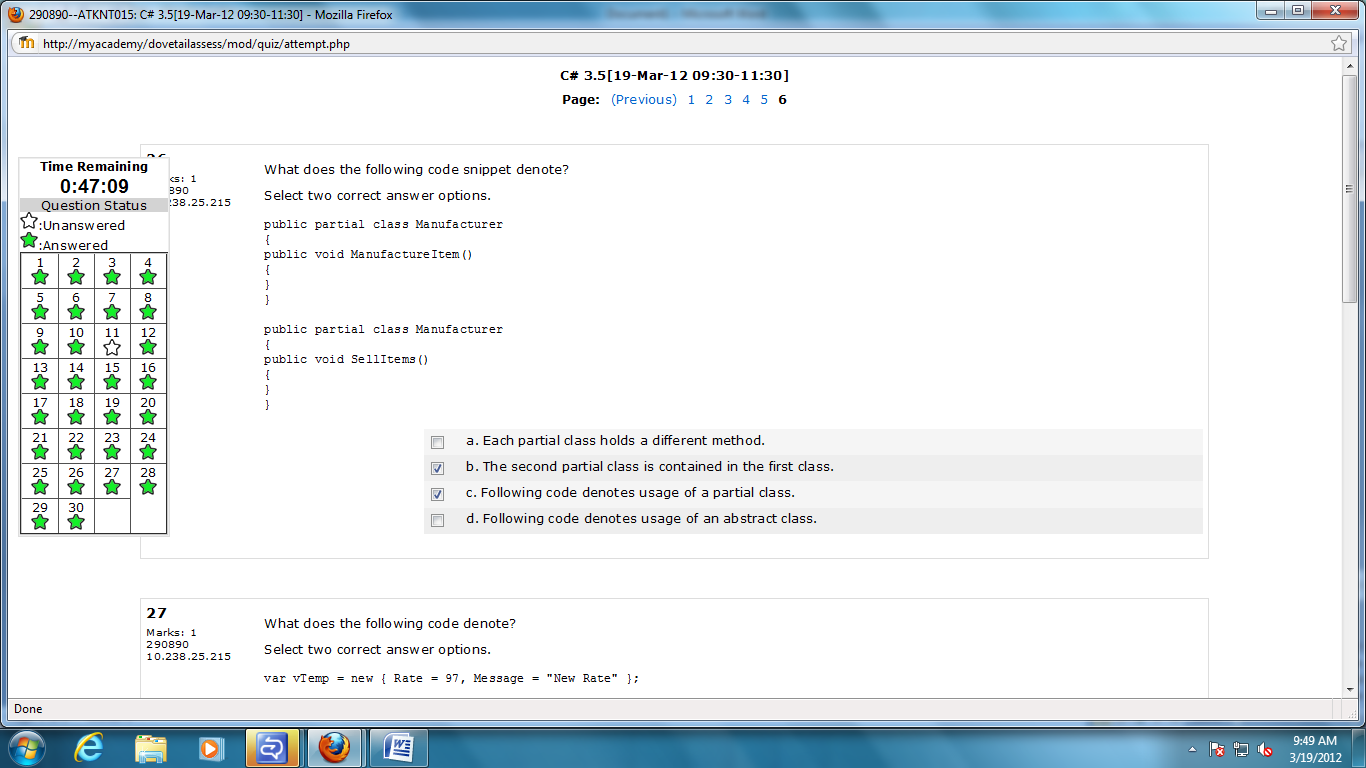


23.b,c,e

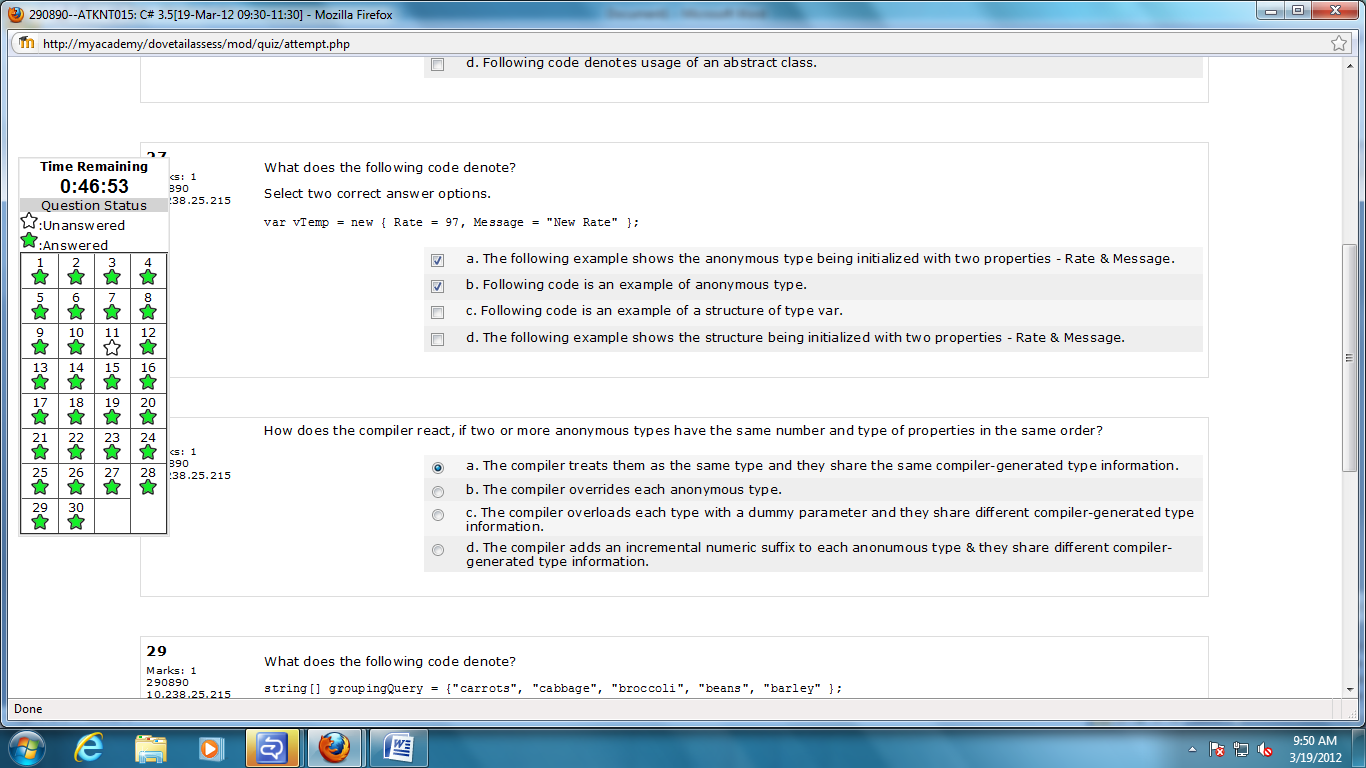
24.a,e



25.d

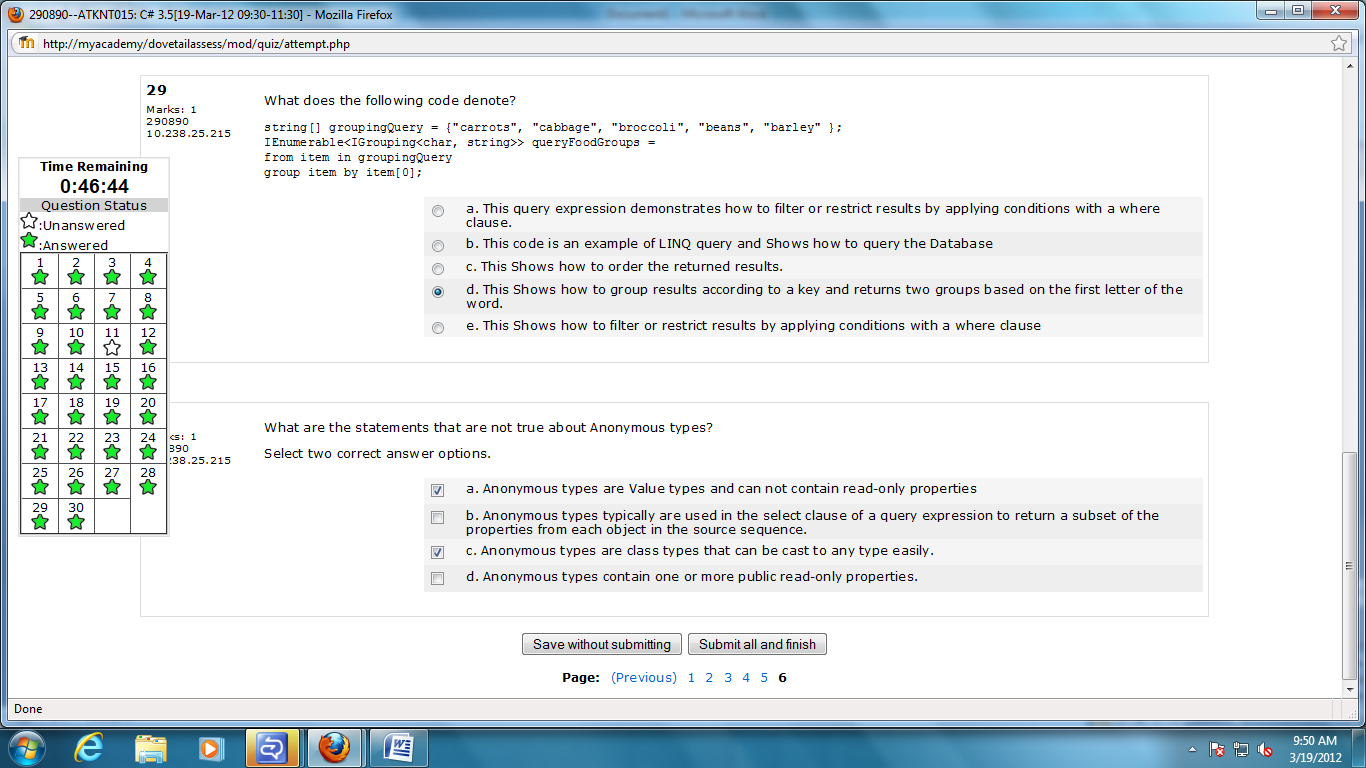


26.a,c



27.a,b

28a



29.d

30.a,c