

Department of Master of Computer Applications



.NET LABORATORY MANUAL

(16MCA57)

- 1. Write a Program in C# to demonstrate Command line arguments processing for the following:**
- a) To find the square root of a given number.**
 - b) To find the sum & average of three numbers.**

Program a)

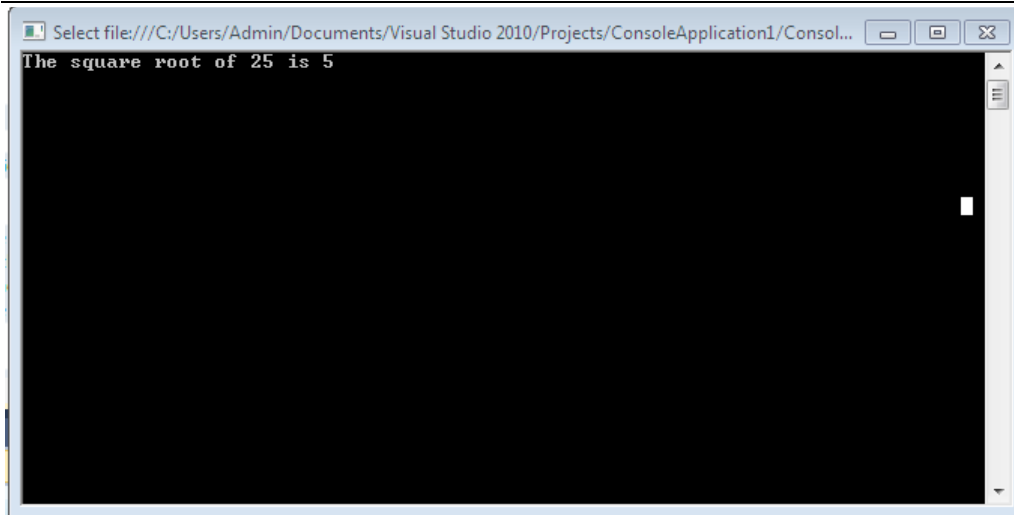
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace LabManual
{
    class Program
    {
        static void Main(string[] args)
        {
            //Type t = sree[0].GetType;
            Double x;

            if (Double.TryParse(args[0], out x))
            {
                double ans = Math.Sqrt(x);
                Console.WriteLine("The square root of " + args[0] + " is " + ans);
            }
            else
                Console.WriteLine("Command line argument should be an integer>=0");

            Console.ReadLine();
        }
    }
}
```

Output



Program b)

using System;

```
namespace LabManual2
{
```

```
    class Program
    {
```

```
        static void Main(string[] sree)
        {
```

```
            double num1, num2, num3;
```

```
            if (Double.TryParse(sree[0], out num1)
&&Double.TryParse(sree[1], out num2) &&Double.TryParse(sree[2], out
num3))
```

```
            {
```

```
                double sum = num1 + num2 + num3;
```

```
Console.WriteLine("Sum of the numbers:" + sum);
```

```
Console.WriteLine("Average of the numbers:" + sum / 3);
```

```
            }
```

```
            else
```

```
Console.WriteLine("All arguments need to be numbers");
```

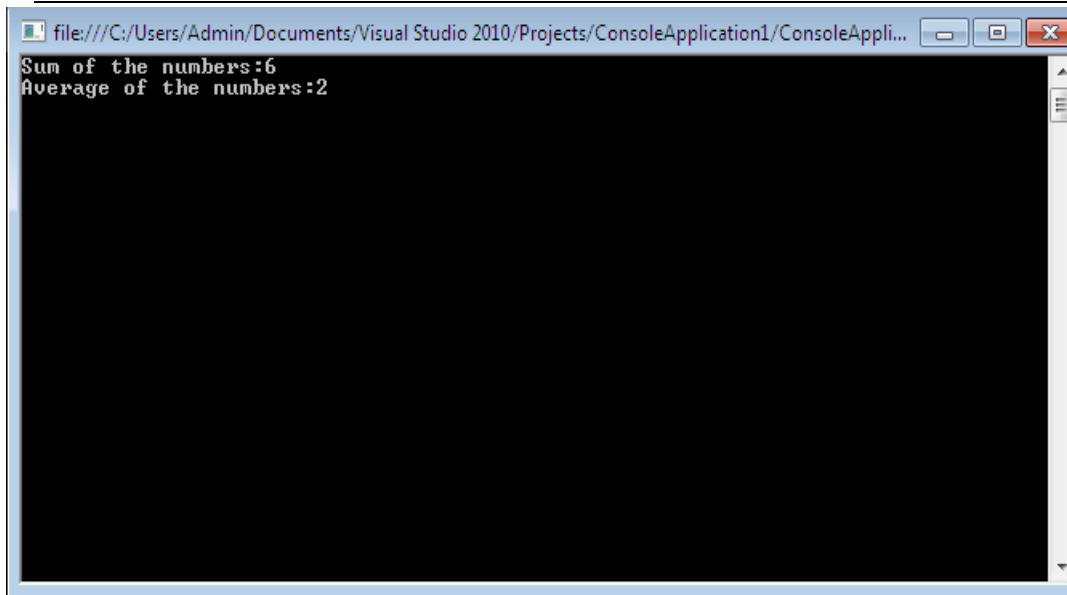
```
Console.ReadLine();
```

```
        }
```

```
    }
```

```
}
```

Output



2. Write a Program in C# to demonstrate the following: a) Boxing and Unboxing b) Invalid Unboxing.

Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

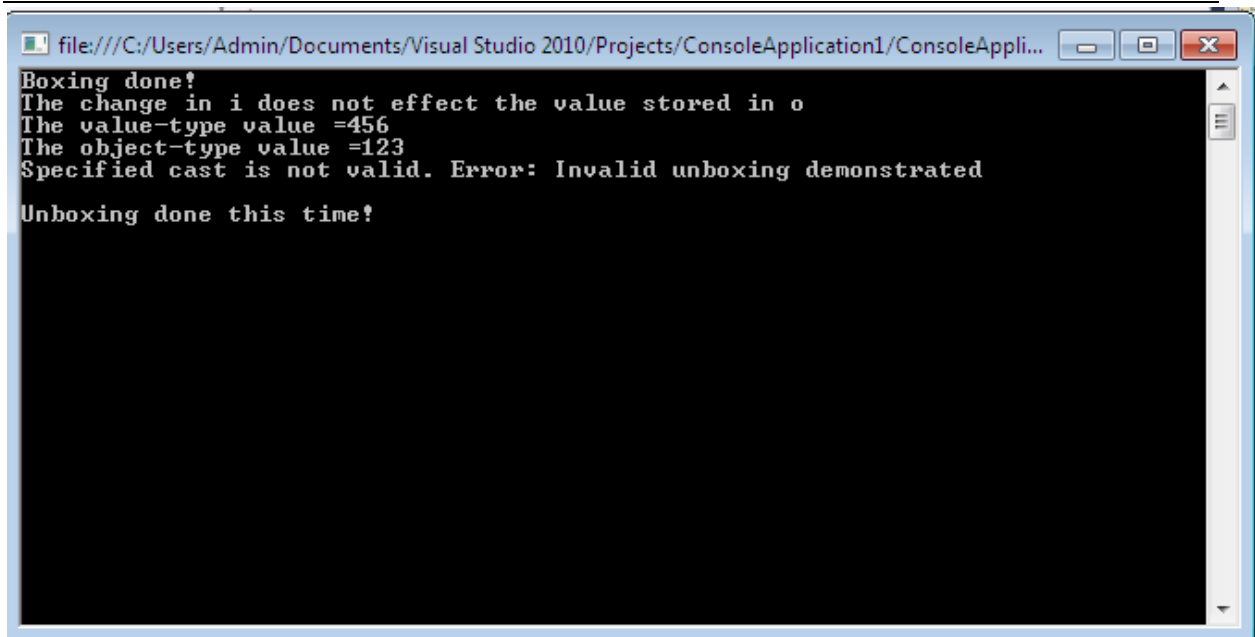
namespace Arjun
{
    class Program
    {
        static void Main(string[] sree)
        {
            int i = 123;

            // Boxing copies the value of i into object o.
            object o = i;
            Console.WriteLine("Boxing done!");

            // Change the value of i.
            i = 456;
```

```
        Console.WriteLine("The change in i does not effect the  
        value stored in o");  
Console.WriteLine("The value-type value =" + i);  
Console.WriteLine("The object-type value =" + o);  
  
        try  
        {  
            int x = (short)o; // attempt to unbox  
System.Console.WriteLine("Unboxing done!");  
        }  
  
        catch (System.InvalidCastException e)  
        {  
            System.Console.WriteLine("{0} Error: Invalid unboxing  
            demonstrated", e.Message);  
            int x = (int)o; // attempt to unbox  
            System.Console.WriteLine("\nUnboxing done this  
            time!");  
        }  
  
Console.ReadLine();  
    }  
}
```

Output



```
file:///C:/Users/Admin/Documents/Visual Studio 2010/Projects/ConsoleApplication1/ConsoleAppli...  
Boxing done!  
The change in i does not effect the value stored in o  
The value-type value =456  
The object-type value =123  
Specified cast is not valid. Error: Invalid unboxing demonstrated  
Unboxing done this time!
```

3. Write a program in C# to add two complex numbers using Operator overloading.**Program**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Arjun
{
    public class Complex
    {
        public int real;
        public int imaginary;

        public Complex(int real, int imaginary)
        {
            this.real = real;
            this.imaginary = imaginary;
        }

        public static Complex operator +(Complex c1, Complex c2)
        {
            return new Complex(c1.real + c2.real, c1.imaginary +
c2.imaginary);
        }

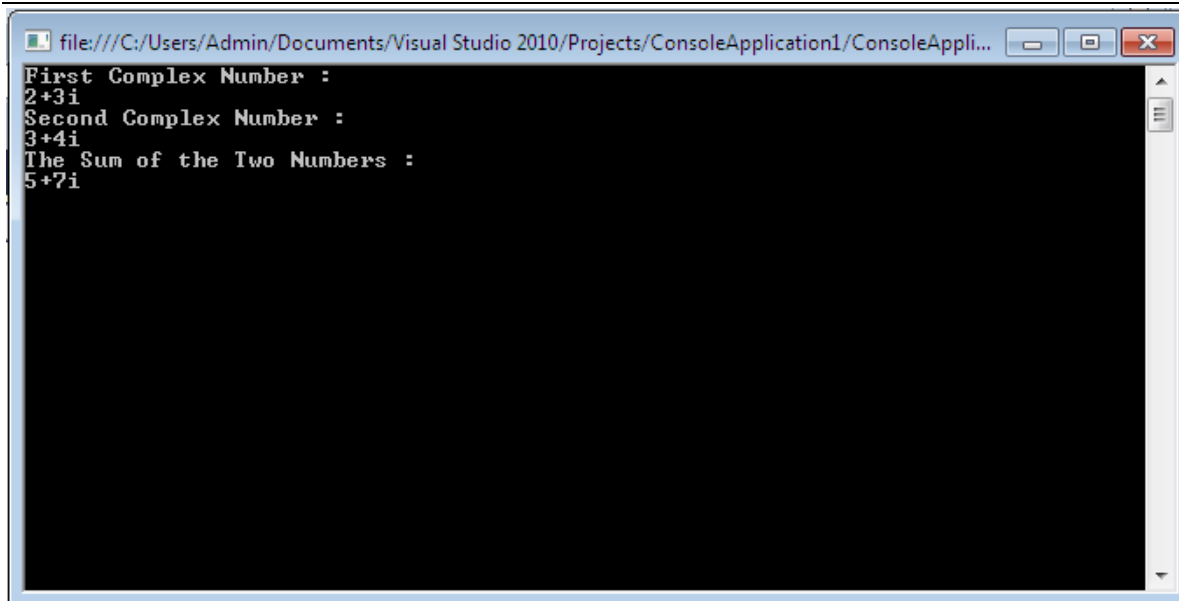
        public void show()
        {
            Console.WriteLine(real+" "+imaginary+"i");
        }
    }

    public class Prog3
    {
        public static void Main()
        {
            Complex num1 = new Complex(2, 3);
            Console.WriteLine("First Complex Number : ");
            num1.show();
            Complex num2 = new Complex(3, 4);
            Console.WriteLine("Second Complex Number : ");
            num2.show();
            Complex sum = num1 + num2;
            Console.WriteLine("The Sum of the Two Numbers : ");
        }
    }
}

```

```
sum.show();  
Console.ReadLine();  
    }  
}  
}
```

Output



4. Find the sum of all the elements present in a jagged array of 3 inner arrays.

Program

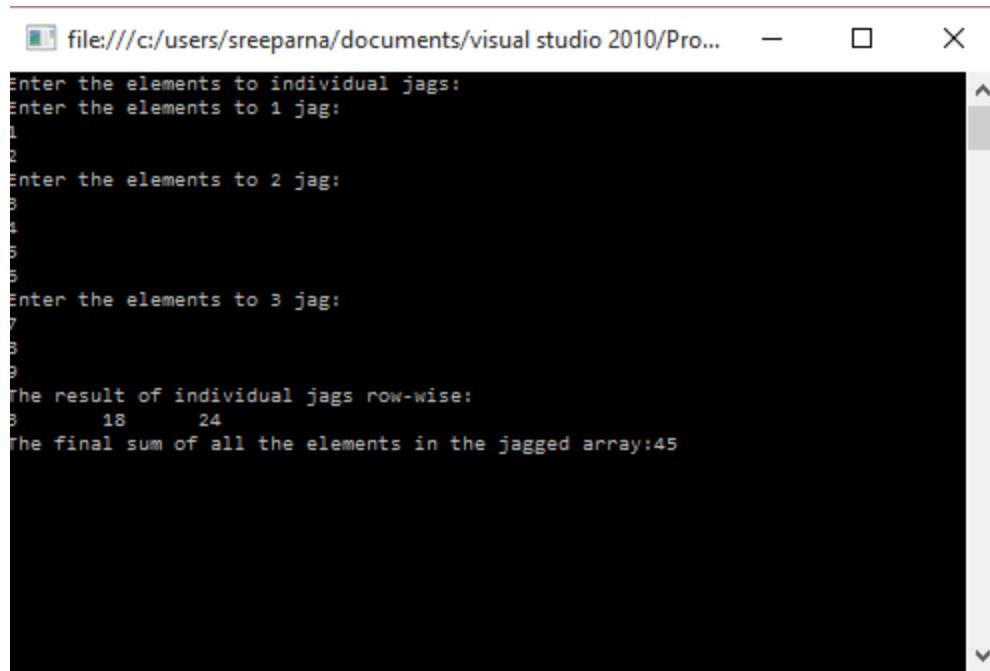
```
using System;  
using System.Collections.Generic;  
using System.Linq;  
using System.Text;
```

```

namespace Arjun
{
    public class Sree
    {
        public static void Main()
        {
            int[][] jag = new int[3][];
            jag[0] = new int[2];
            jag[1] = new int[4];
            jag[2] = new int[3];
            int[] sum = new int[3];
            int result = 0;
            int i, j;
            for (i = 0; i < 3; i++)
                sum[i] = 0;
            Console.WriteLine("Enter the elements to individual jags: ");
            for (i = 0; i < jag.Length; i++)
            {
                Console.WriteLine("Enter the elements to " + (i + 1) + " jag:");
                for (j = 0; j < jag[i].Length; j++)
                {
                    jag[i][j] = int.Parse(Console.ReadLine());
                }
            }
            for (i = 0; i < jag.Length; i++)
            {
                for (j = 0; j < jag[i].Length; j++)
                {
                    sum[i] = sum[i] + jag[i][j];
                }
            }
            for (i = 0; i < 3; i++)
                result = result + sum[i];
            Console.WriteLine("The result of individual jags row-wise: ");
            for (i = 0; i < 3; i++)
                Console.Write(sum[i] + "\t");
            Console.WriteLine();
            Console.WriteLine("The final sum of all the elements in the jagged array:" + result);
            Console.Read();
        }
    }
}

```


Output



```
file:///c:/users/sreeparna/documents/visual studio 2010/Pro...
Enter the elements to individual jags:
Enter the elements to 1 jag:
1
2
Enter the elements to 2 jag:
3
4
5
Enter the elements to 3 jag:
6
7
8
The result of individual jags row-wise:
3      18      24
The final sum of all the elements in the jagged array:45
```

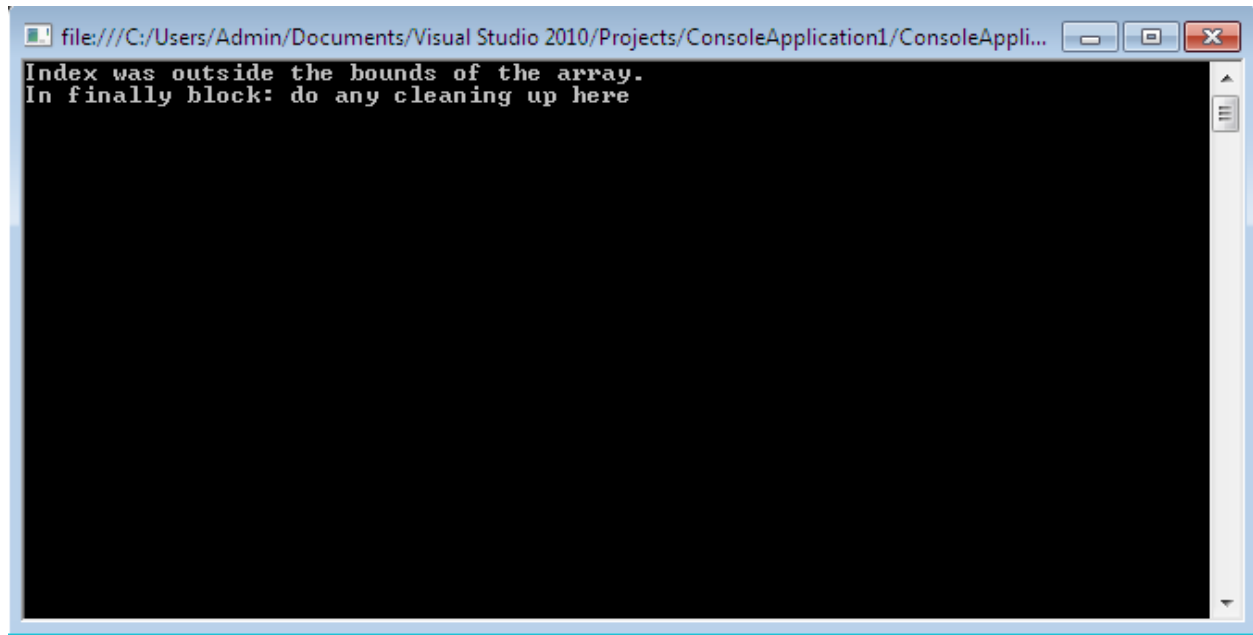
5. Write a Program in C# to demonstrate Array Out of Bound Exception using Try, Catch and Finally blocks.

Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Arjun
{
    public class Prog3
    {
        public static void Main()
        {
            try
            {
                int[] array = new int[100];
                array[0] = 1;
                array[10] = 2;
                array[200] = 3;
                Console.WriteLine("Thisline will never be excecuted");
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
            finally
            {
                // code that does any cleaning up
                Console.WriteLine("In finally block: do any cleaning up here");
                Console.ReadKey();
            }
        }
    }
}
```

Output



The screenshot shows a console application window with a title bar that reads "file:///C:/Users/Admin/Documents/Visual Studio 2010/Projects/ConsoleApplication1/ConsoleAppli...". The window contains two lines of text: "Index was outside the bounds of the array." followed by "In finally block: do any cleaning up here". The text is displayed in a monospaced font on a black background.

```
file:///C:/Users/Admin/Documents/Visual Studio 2010/Projects/ConsoleApplication1/ConsoleAppli...
Index was outside the bounds of the array.
In finally block: do any cleaning up here
```

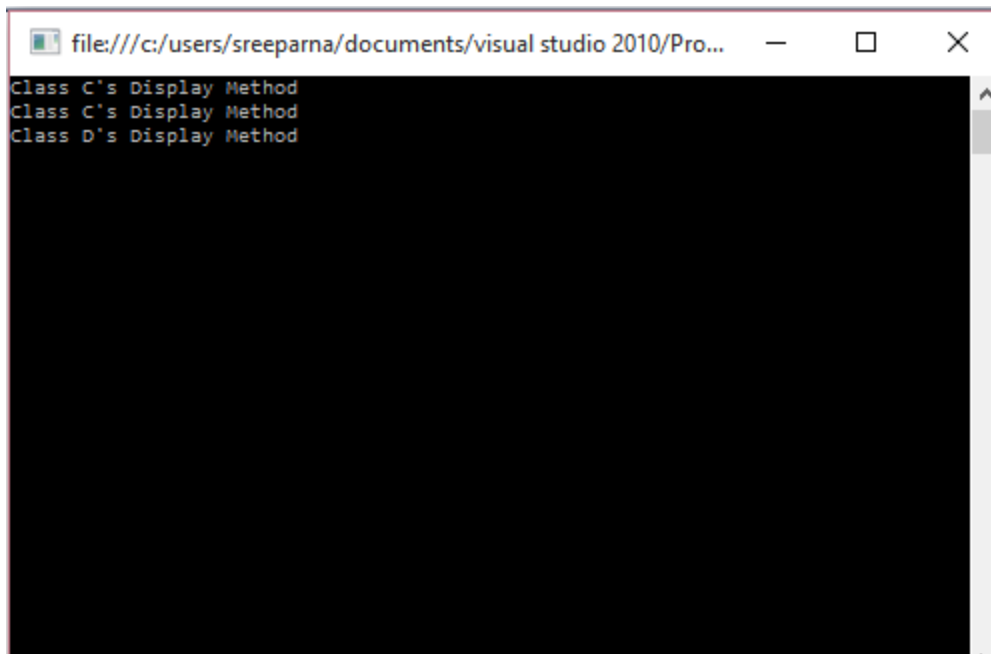
6. Demonstrate Use of Virtual and override key words in C# with a simple program.**Program**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Arjun
{
    public class Prog6
    {
        static void Main(string[] sree)
        {
            //The function called is based
            //upon the type called by new.
            B MyB = new C();
            C MyC = new D();
            D MyD = new D();
            MyB.Display();    //Calls C Display
            MyC.Display();    //Calls C Display
            MyD.Display();    //Calls D Display
            Console.ReadKey();
        }
        public interface A
        {
            void Display();
        }
        class B : A
        {
            public virtual void Display()
            {
                Console.WriteLine("Class B's Display Method");
            }
        }
        class C : B
        {
            public override void Display()
            {
                Console.WriteLine("Class C's Display Method");
            }
        }
    }
}
```

```
    }  
    class D : C  
    {  
        public new void Display()  
        {  
            Console.WriteLine("Class D's Display Method");  
        }  
    }  
}
```

Output



The screenshot shows a console window with the following output:

```
Class C's Display Method  
Class C's Display Method  
Class D's Display Method
```

7. Write a Program in C# to create and implement a Delegate for any two arithmetic operations

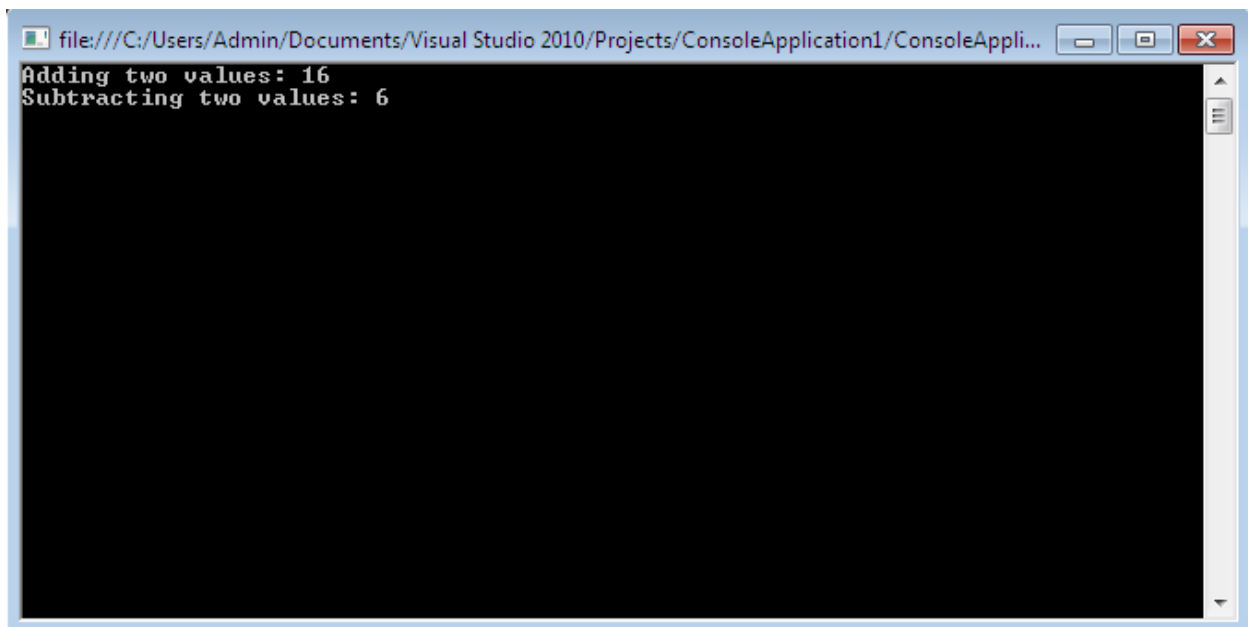
Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Arjun
{
    public delegate int Calculate(int value1, int value2);

    class MyClass
    {
        public int add(int value1, int value2)
        {
            return value1 + value2;
        }
        //a method, that will be assigned to delegate objects must
        //the EXACT return type and parameter list of the delegate
        public static int sub(int value1, int value2)
        {
            return value1 - value2;
        }
        static void Main()
        {
            MyClass obj = new MyClass();
            Calculate del = obj.add;
            Console.WriteLine("Adding two values: " + del(10, 6));
            del = sub;
            Console.WriteLine("Subtracting two values: " + del(10, 4));
            Console.ReadKey();
        }
    }
}
```

Output



The screenshot shows a Windows-style window titled "file:///C:/Users/Admin/Documents/Visual Studio 2010/Projects/ConsoleApplication1/ConsoleAppli...". The window contains a black console area with white text. The first line of text is "Adding two values: 16" and the second line is "Subtracting two values: 6". A vertical scrollbar is visible on the right side of the console area.

```
file:///C:/Users/Admin/Documents/Visual Studio 2010/Projects/ConsoleApplication1/ConsoleAppli...  
Adding two values: 16  
Subtracting two values: 6
```

8. Write a program to demonstrate abstract class and abstract methods in C#.**Program**

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Arjun
{
    abstract public class MotorVehicle
    {
        public string make;
        public string model;

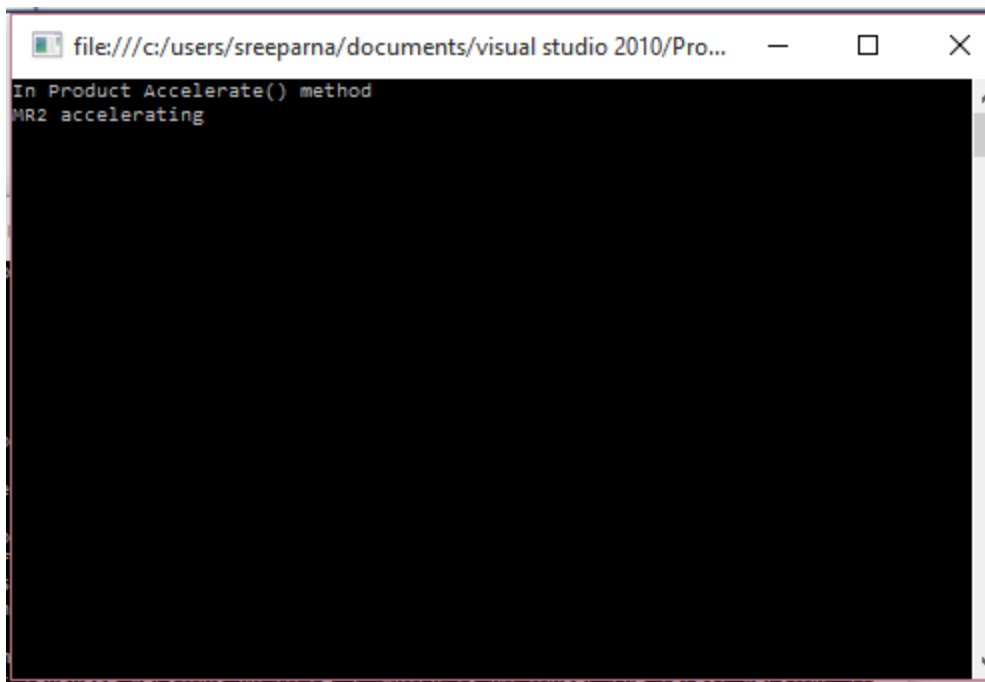
        public MotorVehicle(string make, string model)
        {
            this.make = make;
            this.model = model;
        }
        abstract public void Accelerate();
    }

    public class Product :MotorVehicle
    {
        public Product(string make, string model) :
        base(make, model)
        {
            // do nothing
        }
        public override void Accelerate()
        {
            Console.WriteLine("In Product Accelerate() method");
            Console.WriteLine(model + " accelerating");
        }
    }
}
```



```
class Prog8
{
    public static void Main()
    {
        Product myProduct = new Product("Toyota", "MR2");
myProduct.Accelerate();
Console.ReadLine();
    }
}
```

Output



9. Write a program to Set & Get the Name & Age of a person using Properties of C# to illustrate the use of different properties in C#.

Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

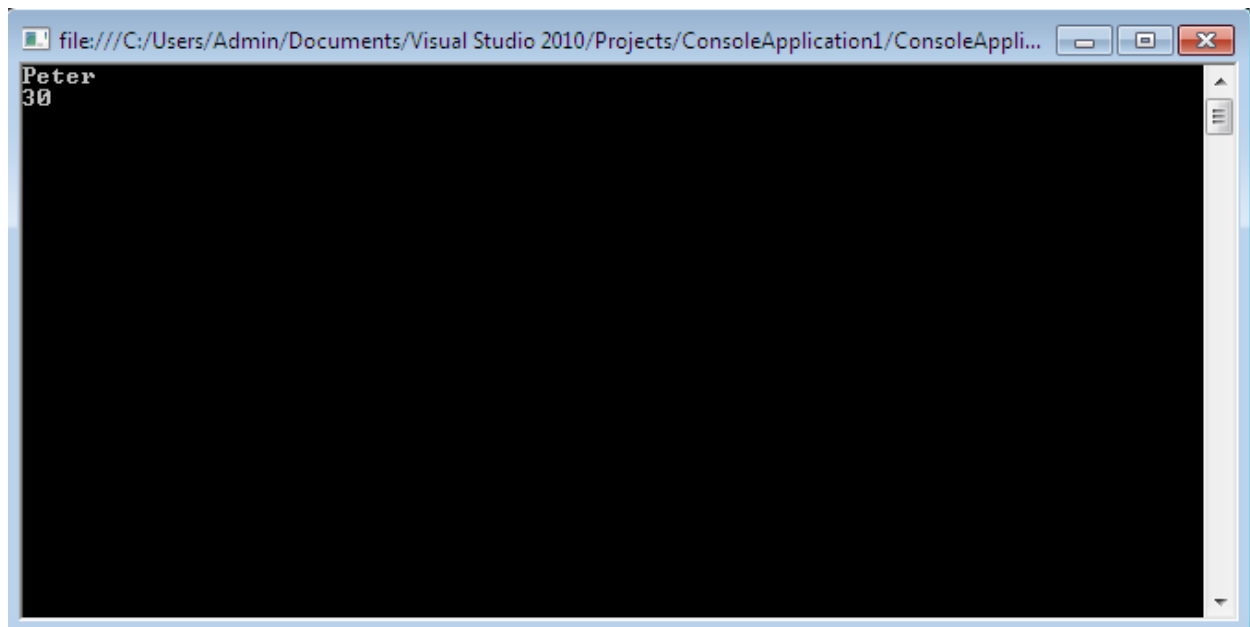
namespace Arjun
{
    public class PropertyClass
    {
        private string name;
        private int age;

        public string p_name
        {
            get
            {
                return name;
            }
            set
            {
                name = value;
            }
        }

        public int p_age
        {
            get
            {
                return age;
            }
            set
            {
                age = value;
            }
        }
    }
}
```

```
        {  
            age = value;  
        }  
    }  
  
    public class MainClass  
    {  
        static void Main(string[] args)  
        {  
PropertyClass ob1 = new PropertyClass();  
            ob1.p_name = "Peter";  
Console.WriteLine(ob1.p_name);  
PropertyClass ob2 = new PropertyClass();  
            ob2.p_age = 30;  
Console.WriteLine(ob2.p_age);  
  
Console.ReadLine();  
        }  
    }  
}
```

Output



10. Write a Program in C# to demonstrate arrays of interface types (for runtime polymorphism).**Program**

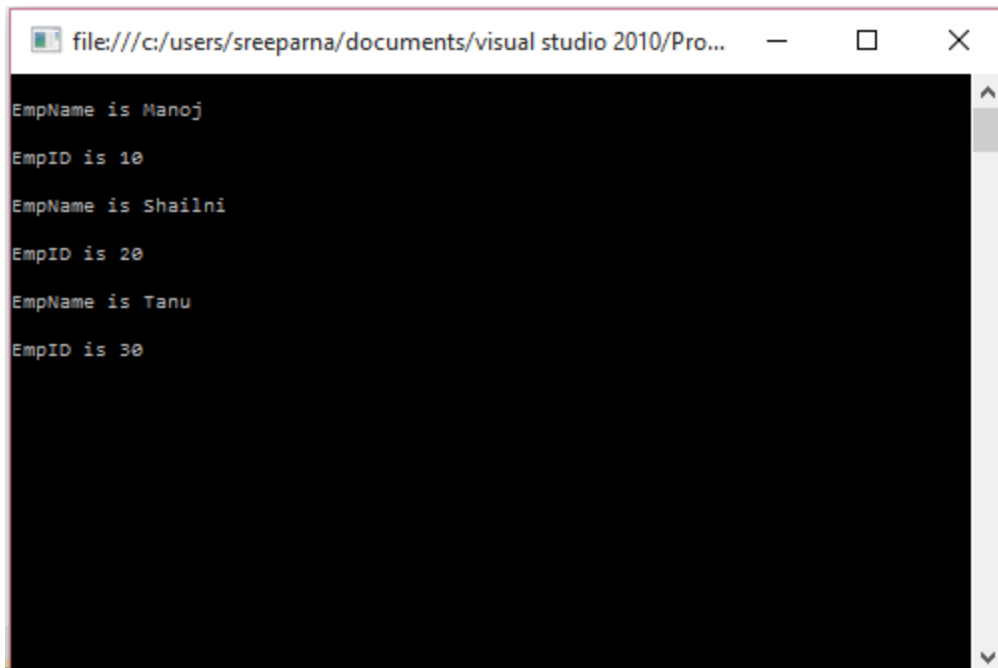
```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;

interface Emp
{
    void EmpInfo();
}
class Employee : Emp
{
    string name;
    int id;
    public Employee(string name, int id)
    {
        this.name = name;
        this.id = id;
    }
    public void EmpInfo()
    {
        Console.WriteLine("\nEmpName is {0}", name);
        Console.WriteLine("\nEmpID is {0}", id);
    }
}
public class Prog10
{
    public static void Main()
```

```
{
Emp[] emps = new Employee[3];
emps[0] = new Employee("Manoj", 10);
emps[1] = new Employee("Shailni", 20);
emps[2] = new Employee("Tanu", 30);
    for (int i = 0; i < 3; i++)
    {
        emps[i].EmpInfo();
    }

Console.ReadLine();
}
}
```

Output



```
file:///c:/users/sreeparna/documents/visual studio 2010/Pro...
EmpName is Manoj
EmpID is 10
EmpName is Shailni
EmpID is 20
EmpName is Tanu
EmpID is 30
```

Sample Viva Questions

1.Does C# support multiple-inheritance?

No. But you can use Interfaces.

2.Where is a protected class-level variable available?

It is available to any sub-class derived from base class

3.Are private class-level variables inherited?

Yes, but they are not accessible.

4.Describe the accessibility modifier “protected internal”.

It is available to classes that are within the same assembly and derived from the specified base class.

6.Which class is at the top of .NET class hierarchy?

System.Object.

7.What does the term immutable mean?

The data value may not be changed. Note: The variable value may be changed, but the original immutable data value was discarded and a new data value was created in memory.

8.What’s the difference between System.String and System.Text.StringBuilder classes?

System.String is immutable. System.StringBuilder was designed with the purpose of having a mutable string where a variety of operations can be performed.

9.What’s the advantage of using System.Text.StringBuilder over System.String?

StringBuilder is more efficient in cases where there is a large amount of string manipulation. Strings are immutable, so each time a string is changed, a new instance in memory is created.

10.Can you store multiple data types in System.Array? No.

11.What's the difference between the System.Array.CopyTo() and System.Array.Clone()?

The Clone() method returns a new array (a shallow copy) object containing all the elements in the original array. The CopyTo() method copies the elements into another existing array. Both perform a shallow copy. A shallow copy means the contents (each array element) contains references to the same object as the elements in the original array. A deep copy (which neither of these methods performs) would create a new instance of each element's object, resulting in a different, yet identical object.

12.How can you sort the elements of the array in descending order?

By calling Sort() and then Reverse() methods.

13.What's the .NET collection class that allows an element to be accessed using a unique key? Hashtable.

14.What class is underneath the SortedList class? A sorted Hashtable.

15.Will the finally block get executed if an exception has not occurred? Yes.

16.What's the C# syntax to catch any possible exception?

A catch block that catches the exception of type System.Exception. You can also omit the parameter data type in this case and just write catch {}.

17.Can multiple catch blocks be executed for a single try statement?

No. Once the proper catch block processed, control is transferred to the finally block.

18.Explain the three services model commonly know as a three-tier application?

Presentation (UI), Business (logic and underlying code) and Data (from storage or other sources).

19.What is the syntax to inherit from a class in C#?

Place a colon and then the name of the base class. Example: class MyNewClass : MyBaseClass

20.Can you prevent your class from being inherited by another class?

Yes. The keyword "sealed" will prevent the class from being inherited.

21.Can you allow a class to be inherited, but prevent the method from being over-ridden?

Yes. Just leave the class public and make the method sealed.

22.What's an abstract class? A class that cannot be instantiated. An abstract class is a class that must be inherited and have the methods overridden. An abstract class is essentially a blueprint for a class without any implementation.

23. When do you absolutely have to declare a class as abstract?

When the class itself is inherited from an abstract class, but not all base abstract methods have been overridden.

and

When at least one of the methods in the class is abstract.

24. What is an interface class? Interfaces, like classes, define a set of properties, methods, and events. But unlike classes, interfaces do not provide implementation. They are implemented by classes, and defined as separate entities from classes.

25. Why can't you specify the accessibility modifier for methods inside the interface?

They all must be public, and are therefore public by default.

26. Can you inherit multiple interfaces? Yes. .NET does support multiple interfaces.

27. What happens if you inherit multiple interfaces and they have conflicting method names?

It's up to you to implement the method inside your own class, so implementation is left entirely up to you. This might cause a problem on a higher-level scale if similarly named methods from different interfaces expect different data, but as far as compiler cares you're okay.

28. What's the difference between an interface and abstract class?

In an interface class, all methods are abstract - there is no implementation. In an abstract class some methods can be concrete. In an interface class, no accessibility modifiers are allowed. An abstract class may have accessibility modifiers.

29. What is the difference between a Struct and a Class? Structs are value-type variables and are thus saved on the stack, additional overhead but faster retrieval. Another difference is that structs cannot inherit.

30. What's the implicit name of the parameter that gets passed into the set method/property of a class? Value. The data type of the value parameter is defined by whatever data type the property is declared.

31. What does the keyword "virtual" declare for a method or property?

The method or property can be overridden.

32. How is method overriding different from method overloading? When overriding a method, you change the behavior of the method for the derived class. Overloading a method simply involves having another method with the same name within the class.

33. Can you declare an override method to be static if the original method is not static?

No. The signature of the virtual method must remain the same. (Note: Only the keyword virtual is changed to keyword override)

34. What are the different ways a method can be overloaded?

Different parameter data types, different number of parameters, different order of parameters.

35. If a base class has a number of overloaded constructors, and an inheriting class has a number of overloaded constructors; can you enforce a call from an inherited constructor to a specific base constructor? Yes, just place a colon, and then keyword base (parameter list to invoke the appropriate constructor) in the overloaded constructor definition inside the inherited class.

36. What's a delegate? A delegate object encapsulates a reference to a method.

37. What's a multicast delegate?

A delegate that has multiple handlers assigned to it. Each assigned handler (method) is called.

38. What's the implicit name of the parameter that gets passed into the class' set method?

Value, and it's datatype depends on whatever variable we're changing.

39. How do you inherit from a class in C#? Place a colon and then the name of the base class.

40. Does C# support multiple inheritance? No, use interfaces instead.

41. When you inherit a protected class-level variable, who is it available to?

Classes in the same namespace.

42. Are private class-level variables inherited?

Yes, but they are not accessible, so looking at it you can honestly say that they are not inherited.

43. Describe the accessibility modifier protected internal. It's available to derived classes and classes within the same Assembly (and naturally from the base class it's declared in).

44. C# provides a default constructor for me. I write a constructor that takes a string as a parameter, but want to keep the no parameter one. How many constructors should I write? Two. Once you write at least one constructor, C# cancels the freebie constructor, and now you have to write one yourself, even if there's no implementation in it.

45. What's the top .NET class that everything is derived from? System.Object.

46. How's method overriding different from overloading? When overriding, you change the method behavior for a derived class. Overloading simply involves having a method with the same name within the class.

47. What does the keyword virtual mean in the method definition? The method can be over-ridden.

48. Can you declare the override method static while the original method is non-static?

No, you can't, the signature of the virtual method must remain the same, only the keyword virtual is changed to keyword override.

49. Can you override private virtual methods? No, moreover, you cannot access private methods in inherited classes, have to be protected in the base class to allow any sort of access.

50. Can you prevent your class from being inherited and becoming a base class for some other classes? Yes, that's what keyword sealed in the class definition is for. The developer trying to derive from your class will get a message: cannot inherit from Sealed class Whatever Base Class Name. It's the same concept as final class in Java.

51. Can you allow class to be inherited, but prevent the method from being over-ridden? Yes, just leave the class public and make the method sealed.

52. What's an abstract class? A class that cannot be instantiated. A concept in C++ known as pure virtual method. A class that must be inherited and have the methods over-ridden. Essentially, it's a blueprint for a class without any implementation.

53. When do you absolutely have to declare a class as abstract (as opposed to free-willed educated choice or decision based on UML diagram)? When at least one of the methods in the class is abstract. When the class itself is inherited from an abstract class, but not all base abstract methods have been over-ridden.

54. What's an interface class? It's an abstract class with public abstract methods all of which must be implemented in the inherited classes.

55. Why can't you specify the accessibility modifier for methods inside the interface? They all must be public. Therefore, to prevent you from getting the false impression that you have any freedom of choice, you are not allowed to specify any accessibility, it's public by default.

56. Can you inherit multiple interfaces? Yes, why not.

57. And if they have conflicting method names?

It's up to you to implement the method inside your own class, so implementation is left entirely up to you. This might cause a problem on a higher-level scale if similarly named methods from different interfaces expect different data, but as far as compiler cares you're okay.

58. What's the difference between an interface and abstract class?

In the interface all methods must be abstract, in the abstract class some methods can be concrete. In the interface no accessibility modifiers are allowed, which is ok in abstract classes.

59. How can you overload a method?

Different parameter data types, different number of parameters, different order of parameters.

60. If a base class has a bunch of overloaded constructors, and an inherited class has another bunch of overloaded constructors, can you enforce a call from an inherited constructor to an arbitrary base constructor? Yes, just place a colon, and then keyword base

(parameter list to invoke the appropriate constructor) in the overloaded constructor definition inside the inherited class.

61. What's the difference between System.String and System.StringBuilder classes?

System.String is immutable, System.StringBuilder was designed with the purpose of having a mutable string where a variety of operations can be performed.

62. Is it namespace class or class namespace? The .NET class library is organized into namespaces. Each namespace contains a functionally related group of classes so natural namespace comes first.