C# Programming With .NET
(06CS/IS761)

Chapter wise questions appeared in previous years:

UNIT III:	**C# Language Fundamentals**	Markes & Year Appeared
1	Why System.Object is called master node? List and explain any three instance methods and static methods of System.Object. Or Why System.Object is called master node? List and explain any three instance methods and static methods of System.Object. Or What is the role of master node System.Object? Or List the methods in System.Object master node. Explain the functionality of the methods Equals(), ToString(), GetType().	June 12 (08m) Dec- 10 (10M) Dec- 09 (07M)

Ans

- It is known that every data type(ValueType or Reference type) is ultimately derived from a common base class: System.Object. Thus it is known as Master Node.
- The Object class defines a common polymorphic behavior for every type in the .NET universe.
- Implicitly, It identifies the base class and derived classes and makes the difference between them.

The topmost class in the .NET world: System.Object

```
namespace System
{
Public class Object
  {
        public Object();
        public virtual Boolean Equals(Object obj);
        public virtual Int32 GetHashCode();
        public Type GetType();
        public  virtual String ToString();
        protected        virtual void Finalize();
        protected Object MemberWiseClone();
        public static bool Equals(object objA, object objB);
        public static bool ReferenceEquals(object objA, object objB);
        }
}
```

Instance Method of Object Class	**Meaning in Life:** **ObjTest c1 = new ObjTest(); ObjTest c2 = c1; object o = c2;**
Equals()	By default this method returns true value only if the items being compared refer to the exact same item in the memory.Thus Equals() is used to compare object references. e. g.: if(c1.Equals(c) && c2.Equals(o)), Wrt.Ln("Same Instances"); Instances");
GetHash Code()	Returns the integer value that identifies a specific object instance. E. g.:
GetType()	This methods returns a System.Type object that fully desribes details of the type you are currently referencing. e. g.:
ToString()	Returns the string representation of the given object, using the <namespace <Class Name>> format.e.g.:
Finalize()	It's a protected method invoked by the .NET runtime when an

		object is to be removed from the heap
	Member WiseClone()	This protected method exist to return a new object that is a member- by- member copy of the current object. Thus, If your object contains reference to other objects, the *reference* to these types are copied.

2 With an example, Explain what happens when reference type is passed by value and passed by reference. June 12 (04m)

Or

With an illustrative example, Explain what happens when reference type is passed by value and when reference type is passed by reference. Dec-10 (05M)

Or

Distinguish between value types and reference types with an example. Dec-09 (07M)

Ans

Reference type is passed by value:

```
class PassingRefByVal
{
    static void Change(int[] pArray)
    {
        pArray[0] = 888;  // This change affects the original element.
        pArray = new int[5] {-3, -1, -2, -3, -4};   // This change is local.
        System.Console.WriteLine("Inside the method, the first element is: {0}", pArray[0]);
    }

    static void Main()
    {
        int[] arr = {1, 4, 5};
        System.Console.WriteLine("Inside Main, before calling the method, the first element is:
{0}", arr [0]);

        Change(arr);
        System.Console.WriteLine("Inside Main, after calling the method, the first element is:
{0}", arr [0]);
    }
}
/* Output:
    Inside Main, before calling the method, the first element is: 1
    Inside the method, the first element is: -3
    Inside Main, after calling the method, the first element is: 888
*/
```

Reference Type passed by reference:

```
class PassingRefByRef
{
    static void Change(ref int[] pArray)
    {
        // Both of the following changes will affect the original variables:
        pArray[0] = 888;
        pArray = new int[5] {-3, -1, -2, -3, -4};
        System.Console.WriteLine("Inside the method, the first element is: {0}", pArray[0]);
    }

    static void Main()
    {
        int[] arr = {1, 4, 5};
        System.Console.WriteLine("Inside Main, before calling the method, the first element is:
{0}", arr[0]);
```

```
        Change(ref arr);
        System.Console.WriteLine("Inside Main, after calling the method, the first element is: {0}", arr[0]);
    }
}
/* Output:
   Inside Main, before calling the method, the first element is: 1
   Inside the method, the first element is: -3
   Inside Main, after calling the method, the first element is: -3
*/
```

3 Explain boxing and unboxing, with an example. June 12 (06m)

Or

What is boxing and unboxing?Explain with an examples. Dec 09 (08M)

Or

Explain Boxing and Unboxing with examples. June-July11 (06M)

Ans

- We all know that .NET defines two broad categories of types:
  - Value- based
  - Reference- based.
- Occasionally you may need to represent a variable of one category as a variable of the other category.
- Thus, C# provides a special mechanism to convert between value types and reference types, known as "***BOXING".***
- ***Boxing*** can be formally defined as the process of explicitly converting a value type into a corresponding reference type.
- ***Boxing*** a value is like allocating a new object on the heap and copying the internal val into that instance.
- ***UNBOXING*** is the term given to the process of converting the value held in the object reference back into a corresponding value type on the stack.
- Unboxing operation begins by verifying that the receiving data type is equivalent to th boxed type.
- If so, then copy the value out of the box into a local stack based variable.
- // Make short value type:
  short s = 25;
- Consider, during the course of application, if you wish to represent this value types as a reference type, you would "BOX" the values:
- //Box the value into an object reference.
  Object objShort = s;
- //Unboxing the reference back into a corresponding short:
  short another Short = (short)objShort;
- Note: It is mandatory that you need to unbox the value into an appropriate data type.

```
// Program for Boxing……
class Program
{
static void Main(string[] args)
{ // create an int (value type)
Int MyInt = 99;
//Because myInt is passed into a method prototyped to take an object,MyInt is "boxed" Automatically.
UseThisObject(MyInt);
Console.ReadLine();
}
Static void UseThisObject(object o)
{
Console.WriteLine("Value of 0 is {0}", o);
}
```

```
}
//Program to represent Unboxing…….
using System;
class Program
{
   public static void Main(string[] args)
   { //Box ints into Array List
      ArrayList myInts = new ArrayList();
      myInts.Add(88);
      myInts.Add(3.33);
      myInts.Add(false);
      //Unboxing first item from ArryList
      int FirstItem = (int)myInts[0];
      Console.WriteLine("First item is {0}", FirstItem);
   }
}
```

4 Explain four method parameter modifiers, with an example. June 12 (08M)

Or

Explain the method parameter modifiers. Demonstrate with a function definition and function call for each modifier. Dec 11 (10m)

Or

What are the method parameter modifiers? Explain any two C# method parameter modifiers with an example. Dec- 10 (05M)

Ans

Method Parameter Modifiers:

Parameter Modifier	Meaning in Life
(None)	If a parameter is not marked with a parameter modifier, it is assumed to be passed by value, meaning the called method receives a copy of the original data.

```
using System;
class Parameters
{
// Arguments are passed by value by default.
static int Add(int x, int y)
{
int ans = x + y;
// Caller will not see these changes // as you are modifying a copy of the //
original data.
x = 10000; y = 88888;
return ans;
}
static void Main(string[] args)
{
Console.WriteLine("***** Fun with Methods *****");
// Pass two variables in by value.
int x = 9, y = 10;
Console.WriteLine("Before call: X: {0}, Y: {1}", x, y);
Console.WriteLine("Answer is: {0}", Add(x, y));
Console.WriteLine("After call: X: {0}, Y: {1}", x, y);
Console.ReadLine();
}
}
```

Out	Output parameters must be assigned by the method being called (and therefore are passed by reference). If the called method fails to assign output parameters, you are issued a compiler error.

However, the C# out modifier does serve a very useful purpose: it allows the caller to obtain multiple return values from a single method invocation

```
// Returning multiple output parameters.
void static FillTheseValues(out int a, out string b, out bool c)
{
a = 9;
b = "Enjoy your string.";
c = true;
}
static void Main(string[] args)
{
Console.WriteLine("***** Fun with Methods *****");
...
int i; string str; bool b;
FillTheseValues(out i, out str, out b);
Console.WriteLine("Int is: {0}", i);
Console.WriteLine("String is: {0}", str);
Console.WriteLine("Boolean is: {0}", b);
Console.ReadLine();
}
```

ref (Reference)

The value is initially assigned by the caller and may be optionally reassigned by the called method (as the data is also passed by reference).
No compiler error is generated if the called method fails to assign a ref parameter.

```
using System;
class RefParameters    {
// Reference parameters.
public static void SwapStrings(ref string s1, ref string s2)     {
string tempStr = s1;
s1 = s2;
s2 = tempStr;
}
static void Main(string[] args)            {
Console.WriteLine("***** Fun with Methods *****");
string s1 = "Flip";
string s2 = "Flop";
Console.WriteLine("Before: {0}, {1} ", s1, s2);
SwapStrings(ref s1, ref s2);
Console.WriteLine("After: {0}, {1} ", s1, s2);
Console.ReadLine(); }
}
```

params (Parameters)

This parameter modifier allows you to send in a variable number of arguments as a single logical parameter.
A method can have only a single params modifier, and it must be the final parameter of the method.

```
//This method has two physical parameters.
Public static void DisplayArrayofInts(string msg, params int[] list)
{
Console.WriteLine(msg);
for(int i= 0; i < list.Length; i++)
Console.WriteLine("list[i]");
}
Two Physical parameters:
```

```
String type,
parameterized array of integers.
This method is saying like, “send me a string as the first parameter and any number
of integer as second”.
//use ‘params‘ keyword modifier:
int[] intArray = new int[3] {10, 11, 12};
DisplayArrayofInts(“Here is an array of ints”, intArray);
DisplayArrayofInts(“Enjoy these 3 ints”, 1, 2, 3);
DisplayArrayofInts(“Take some more!”,55, 4, 983, 1043, 98, 33);
```

5 Write a C# program to arrange 5 names in ascending order. The names are obtained from the command line arguments. Dec 11 (06M)

Ans

```
using System;
namespace Hello
{
    class HelloClass
    {
        public static int Main(string[] args)
        {
            Console.WriteLine("**CMD Line Args***");
            Array.Sort(args);
            for(int x=0;x<args.Length;x++)
                Console.WriteLine("Arg {0}: {1}", x,args[x]);
            Console.Read();
            return 0;
        }
    }
}
```

6 Explain the functions of System.Object class. Give overrided definition for ToString() and Equals(). Dec- 11 (10M)

Or

Explain functions of System.Object class. Give overridden definition for ToString() and Equals() member functions. May-June 10 (10M)

June-July11 (08M)

Ans

```
using System;
using System.Text;

class Person
{
    public Person(string fname, string lname, string ssn, byte a)
    {
        firstName = fname;
        lastName  =  lname;
        SSN = ssn;
        age = a;
    }
    public Person() { }
    public string firstName;
    public string lastName;
    public string SSN;
    public byte age;

    //over riding system.Object.ToString()
    public override string ToString()
    {
```

```
        StringBuilder sb = new StringBuilder();
        sb.AppendFormat("[First Name = {0}", this.firstName);
        sb.AppendFormat("Last Name = {0}", this.lastName);
        sb.AppendFormat("SSn = {0}", this.SSN);
        sb.AppendFormat("Age = {0}", this.age);
        return sb.ToString();
    }

class Person   {
public override bool Equals(object o)
{// Does the incoming object instance have the same values as me?
            Person temp = (Person)o;
if(temp.firstName == this.firstName &&
temp.lastName == this.lastName &&
temp.SSN == this.SSn &&
temp.age == this.age)
return true;
else
            return false;   }
……….
}

public static void Main(string[] args)
        {// Note: We want these tobe identical for testing purpose.
        Person P3 = new Person("Fred","Jones","22-22-222",98);
        Person P4 = new Person("Fred","Jones","22-22-222",98);

        //Should have same hash code and string at this point.
        Console.WriteLine("Hash Code of P3 = {0}",P3.GetHashCode());
        Console.WriteLine("Hash Code of P4 = {0}",P4.GetHashCode());
        Console.WriteLine("String of P3 = {0}",P3.ToString());
        Console.WriteLine("String of P4 = {0}",P4.ToString());

/ /      //Should be equal at this point
//        if(P3.Equals(P4))
//            Console.WriteLine("P3 And P4 Have the same State");
//        else
//            Console.WriteLine("P3 and P4 have the different State");

        //Change Age of P4.
        Console.WriteLine("\n-> Changing the age of P4\n");
        P4.age = 2;

        //No longer equals, Different hash values and string data
        Console.WriteLine("->String of P3 = {0}",P3.ToString());
        Console.WriteLine("->String of P4 = {0}",P4.ToString());
        Console.WriteLine("->Hash Code of P3  = {0}",P3.GetHashCode());
        Console.WriteLine("->Hash Code of P4 = {0}", P4.GetHashCode());
        if(P3.Equals(P4))
            Console.WriteLine("P3 And P4 Have the same State");
        else
            Console.WriteLine("P3 and P4 have the different State");
        }
}
```

7 Explain the params modifier, with suitable example. DEC-11 (04M)

This parameter modifier allows you to send in a variable number of arguments as a single logical parameter.
A method can have only a single params modifier, and it must be the final parameter of the method.

Ans

```
//This method has two physical parameters.
Public static void DisplayArrayofInts(string msg, params int[] list)
{
        Console.WriteLine(msg);
        for(int i= 0; i < list.Length; i++)
                Console.WriteLine("list[i]");
}
```

Two Physical parameters:
String type,
parameterized array of integers.

This method is saying like, "send me a string as the first parameter and any number of integer as second".

```
//use 'params' keyword modifier:
int[] intArray = new int[3] {10, 11, 12};
DisplayArrayofInts("Here is an array of ints", intArray);
DisplayArrayofInts("Enjoy these 3 ints", 1, 2, 3);
DisplayArrayofInts("Take some more!",55, 4, 983, 1043, 98, 33);
```

8 Write a program in C# to read a jagged array and display the sum of all the elements of three inner arrays. Dec- 10 (06M)

Ans

```
using System;
using System.Collections.Generic;
using System.Text;
namespace jaggedArrayExample
{
  class program
  {
    static void Main(string[] args)
    {
      //int ans = 0;
      const int rows = 3;
      //declare the jagged array as 3 Rows high
      int[][] jagArr = new int[rows][];
      //a row with 2 elements
      jagArr[0] = new int[2];
      //a row with 3 elements
      jagArr[1] = new int[3];
      //a row with 4 elements
      jagArr[2] = new int[4];
       //fill some elements of the rows.....
       jagArr[0][1] = 54;
       jagArr[1][0] = 26;
       jagArr[1][1] = 18;
       jagArr[2][0] = 72;
       jagArr[2][3] = 404;

    Console.Write("************Elements of a jagged array***********\n");
     for (int i = 0; i <= 2; i++)
       {
          Console.Write("Length of row {0} is {1} :\t", i, jagArr[i].Length);
         for (int j = 0; j < jagArr[i].Length; j++)
```

```
            Console.Write(jagArr[i][j] + " ");
            Console.WriteLine();
        }

    Console.Write("*******Sum of the inner arrays of a jagged array*********\n");
     for (int i = 0; i <= 2; i++)
     {
      int ans = 0;
      for (int j = 0; j < jagArr[i].Length; j++)
      ans = ans + jagArr[i][j];
      Console.Write("Sum of elements of row {0} is {1} :\t", i, ans);
      Console.WriteLine();
     }
   }
  }
 }
```

9 Write a program in C# to sort an array of student objects having rollno, name and marks in two subjects. (Dec-10 (12M))

-Display the array sorted on names.

-Display the array based on average marks.

Ans Follow the information given in case study of handling Arrays page no 160 and theory of ArrayList from page no 154 from Programming in C# by Balgurusamy.

10 Explain the following types with an example, With reference to C#: (Dec- 09 (10M))

i) foreach, ii) ref, iii) params, iv) verbatim, v) enum

Or

Explain the following terms, With an example, With reference to C#: (June-July11 (06M))

i) foreach, ii) params, iii) verbatim.

- The C# foreach keyword allows you to iterate over all items within an array, without the need to test for the array's upper limit.
- Here are two examples using foreach, one to traverse an array of strings and the other to traverse an array of integers:

```
using System;
class carTypes   {
// Iterate array items using foreach.
public static void Main()   {
string[] carTypes = {"Ford", "BMW", "Yugo", "Honda" };
foreach (string c in carTypes)
Console.WriteLine(c);
int[] myInts = { 10, 20, 30, 40 };
foreach (int i in myInts)
Console.WriteLine(i);
Console.Read();   }
}
```

➢ In addition to iterating over simple arrays, foreach is also able to iterate over system-supplied or user-defined collections.

Enumerator (enum) :

- Enumerations are handy programming constructs that allow you to group name/value pairs.
  - *For* e.g: assume you are creating a video-game application that allows the player to select one of three character categories (Wizard, Fighter, or Thief).

```
// A C# enumeration type.
enum CharacterType
{
    Wizard = 100,
    Fighter = 200,
    Thief = 300
}
```

Verbatim:

- C# introduces a @- quoted string literal notation known as verbatim string. Using the verbatim string you are able to bypass the use of cryptic escape characters:
  - e. g: //The following string is printed verbatim, all \marks are displayed!
  String finalString = 2"\n\tString file: 'C:\csharpProjects\strings\strings.cs'";
  Console.WriteLine(final string);
- The output has been prefixed with "\n\t", as these escape characters are not processed in @- quoted string.
- Also verbatim strings can be used to ignore spaces that flow over multiple lines.

11 Write a program in C# to accept two strings and perform the following operations: Dec- 09 (10M)

i) Copy string 2 to string 3.
ii) Check string 1 ends with "ENGG" or not. If it is true, search character 'a' in string 3.
iii) Insert "VTU" in the string 2 at position 6 and display it.

Ans

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

namespace SearchString
{
   class program
   {
      public void Display()
      {
         bool a = true;
         string str1 = "";
         Console.Write("Enter the string: ");
         str1 = Console.ReadLine();

         string str2 = "";
         Console.Write("Enter another string: ");
         str2 = Console.ReadLine();

         //string copy method
         string str3 = string.Copy(str2);
         Console.WriteLine("String str3 is copied from str2: {0}", str3);

         //Check if the string ends with the sets of characters.....
         Console.WriteLine("String str1: {0}\n ends with ENGG?:{1}\n", str1,
str1.EndsWith("ENGG"));
         if (str1.EndsWith("ENGG")== a)
         {
            if (str3.Contains("A")== a)
            Console.WriteLine("Yes str1 ends with ENGG and str3 contains character
\"A\"\n");
         }
         str2 = str2.Insert(6, "VTU");
```

```
            Console.WriteLine("'VTU ' is inserted in string str2. string s2 is now : {0}\n", str2);
        }
        static void Main(string[] args)
        {
            program prg = new program();
            prg.Display();
        }
    }
}
```

12 Write a C# program to sort and reverse an array of five elements using sort() and reverse Methods. Dec- 09 (04M)

```
using system;
class SortReverse
{
    public static void Main()
    {
        //creating an array
        int[] X = {30, 20, 10, 80, 90, 50};
        Console.WriteLine("Array elements before sorting");
        foreach(int i in X)
        Console.WriteLine(" " + i);
        Console.WriteLine();

        //Sorting and reversing the array elements….
        Array.Sort(X);
        Console.WriteLine("Array after Sorting:\n");
        foreach(int i in X)
        Console.WriteLine(" " +  i);
        Array.Reverse(X);
        Console.WriteLine("Array after sorting and reversing:\n");
        foreach(int i in X)
        Console.WriteLine(" " +  i);
        Console.WriteLine();
    }
}
```

13 Ans Write a C# program to demonstrate use of static and Read- only variables. June-July11 (08M)

Static Variables:

- Static variables are used when we want tohave a variable common to all instances of a class.

```
using System;
public class MathOperation
{
    public static float mul(float x, float y)
    {
        return x * y;
    }
    public static float divide(float x, float y)
    {
        return x / y;
    }
}
class MathApplication
{
    public static void Main()
```

```
    {
        float a = MathOperation.mul(4.0f, 5.0f);
        float b = MathOperaation.divide(a, 2.0f);
        Console.WriteLine("b =" + b);
    }
}
```

Read Only variables:

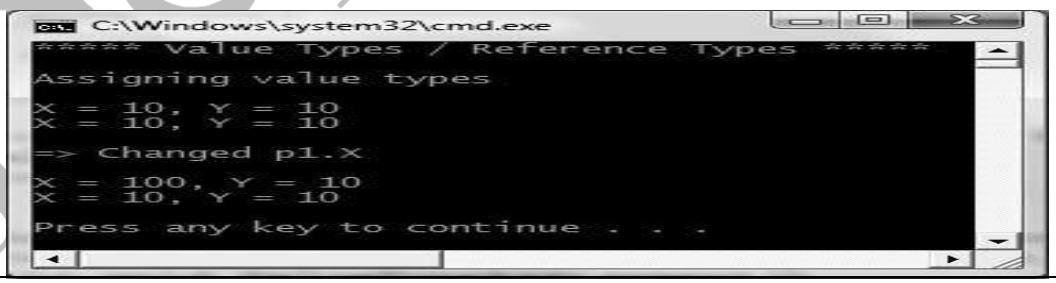
- It is designed to set the value of the member using a constrictor method but cannot be modified later.

```
class Numbers
{
        public readonly int m;
        public static readonly int n;
        public Numbers (int x)
        {
                M = x;
        }
        static Numbers()
        {
                N = 100;
        }
}
```

14 With a program, demonstrate, how an assignment operator, between value types and function types differ.(Refer to page no: 129 to 135 in softcopy of text book) June - 10 (06M)

// Assigning two intrinsic value types results in two independent variables on the stack.

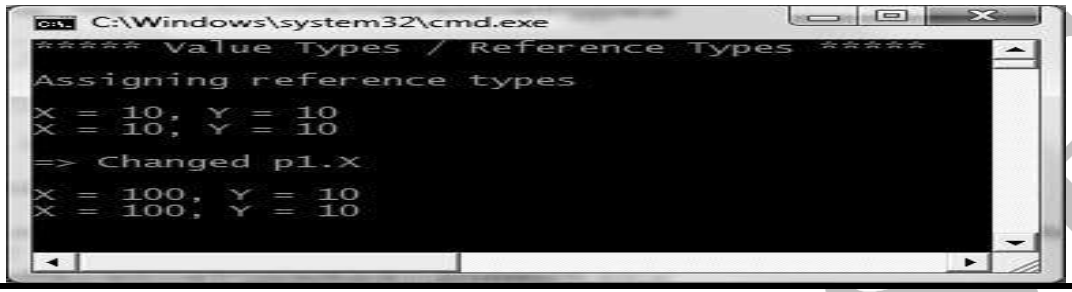
```
static void ValueTypeAssignment()
{
        Console.WriteLine("Assigning value types\n");
        Point p1 = new Point(10, 10);
        Point p2 = p1;
        // Print both points.
        p1.Display();
        p2.Display();
        // Change p1.X and print again. p2.X is not changed.
        p1.X = 100;
        Console.WriteLine("\n=> Changed p1.X\n");
        p1.Display();
        p2.Display();
}
```



- In this case, you have two references pointing to the *same object on the managed heap.*
- *Therefore,* when you change the value of X using the p2 reference, p1.X reports the same value.

```
static void ReferenceTypeAssignment()
{
        Console.WriteLine("Assigning reference types\n");
        PointRef p1 = new PointRef(10, 10);
        PointRef p2 = p1;
```

```
        // Print both point refs.
        p1.Display();
        p2.Display();
        // Change p1.X and print again.
        p1.X = 100;
        Console.WriteLine("\n=> Changed p1.X\n");
        p1.Display();
        p2.Display();
}
```



15 How do you format .NET string and textual output? Give example. Dec- 09 (05M)

Ans

- The C# string keyword is a shorthand notation to represent System.String class Type.
- It provides number of members you would expect from such a utility class.

String Member	Meaning in Life
Length	This property returns the length of the current string.
Compare()	This method compares two strings.
Contains()	This method compares two strings.
Equals()	This method tests whether two string objects contain identical character data.
Copy()	Creates a new string by copying another string
Format()	This method formats a string using other primitives (e.g., numerical data, other strings) and the {0} notation examined earlier in this chapter.
Insert()	This method inserts a string within a given string.
PadLeft(), PadRight()	These methods are used to pad a string with some characters.
Remove(), Replace()	Use these methods to receive a copy of a string, with modifications (characters removed or replaced).
Split()	This method returns a String array containing the substrings in this instance that are delimited by elements of a specified Char or String array.
Trim()	This method removes all occurrences of a set of specified characters from the beginning and end of the current string.
ToUpper(), ToLower()	These methods create a copy of the current string in uppercase or lowercase format, respectively.

```
using System;
class Program
{
    public static void Main(string[] args)
     {
        Console.WriteLine("=> Basic String functionality:");
        string firstName = "Freddy";
        Console.WriteLine("Value of firstName: {0}", firstName);
        Console.WriteLine("firstName has {0} characters.", firstName.Length);
        Console.WriteLine("firstName in uppercase: {0}", firstName.ToUpper());
        Console.WriteLine("firstName in lowercase: {0}", firstName.ToLower());
```

```
            Console.WriteLine("firstName contains the letter y?: {0}",
            firstName.Contains("y"));
            Console.WriteLine("firstName after replace: {0}",firstName.Replace("dy",""));
            Console.WriteLine();
        }
    }
```

```
//Create some objects and exercise the inherited System.Object methods.
using System;
class ObjTest  {
   public static int Main(string[] args)
      {   // Make an instance of the object
       ObjTest c1 = new ObjTest();
       //Pump info into console
       Console.WriteLine("ToString: {0}",c1.ToString());
       Console.WriteLine("Hash Code: {0}",c1.GetHashCode());
       Console.WriteLine("Base Class: {0}",c1.GetType().BaseType);
       // Make some other references to c1
       ObjTest c2 = c1;
       object o = c2;
       // Are all 3instances pointing to the same object in the memory?
       if(o.Equals(c1) && c2.Equals(o))
       Console.WriteLine("Same Instance……!");
      return 0;       }
}
```

In the above program a class ObjTest is created. c1, c2 and 0 are its objects. We are trying to extract the different System.Object method types like: ToString(), GetType(), Equals(), GetHashCode() and checking the same.

16 A simple C# program for Swapping Two strings:

```
class SwappingStrings
 {
    static void SwapStrings(ref string s1, ref string s2)
    // The string parameter is passed by reference.
    // Any changes on parameters will affect the original variables.
    {
       string temp = s1;
       s1 = s2;
       s2 = temp;
       System.Console.WriteLine("Inside the method: {0} {1}", s1, s2);
    }

    static void Main()
    {
       string str1 = "John";
       string str2 = "Smith";
       System.Console.WriteLine("Inside Main, before swapping: {0} {1}", str1, str2);

       SwapStrings(ref str1, ref str2);   // Passing strings by reference
       System.Console.WriteLine("Inside Main, after swapping: {0} {1}", str1, str2);
    }
 }
 /* Output:
    Inside Main, before swapping: John Smith
    Inside the method: Smith John
    Inside Main, after swapping: Smith John
```

```
*/
```

17 A simple C# program to print Alphabets from A to Z in Uppercase and lowercase letters.

```
using System;

namespace SamplePrograms
{
    class Alphabets
    {
        public static void Main()
        {
            // Loop from a thru z (lower case alphabets)
            for (char alphabet = 'a'; alphabet <= 'z'; alphabet++)
            {
                Console.Write(alphabet + " ");
            }
            // Loop from A thru Z (upper case alphabets)
            for (char alphabet = 'A'; alphabet <= 'Z'; alphabet++)
            {
                Console.Write(alphabet + " ");
            }
            Console.ReadLine();
        }
    }
}
```

18 A simple C# program to represent use of Command line execution (CMD line execution). Also is shows the use of foreach keyword too.

```
using System;

namespace Hello
{
    class HelloClass
    {
        public static int Main(string[] args)
        {
        Console.WriteLine("**CMD Line Args***");
        for(int x=0;x<args.Length;x++)
        Console.WriteLine("Arg {0}: {1}", x,args[x]);
        Console.WriteLine("Hello World");
        //Using foreach() keyword for loop execution
                Console.WriteLine("Using the foreach keyword");
        Array.Sort(args);
        foreach(string s in args)
                        Console.WriteLine("Arg: {0}",s);
        Console.Read();
        return 0;
        }
     }
}
```

19 A C# program to represent the use of default and custom Constructors.

```
//HelloClass with Constructors
using System;

class HelloClass
    {
```

```
        //constructors always assign state data to default values
        public HelloClass()
        {
            Console.WriteLine("Default Constructor Called…..!");
        }

    public HelloClass (int x, int y)
    {
    Console.WriteLine("Custom constructor Called");
    intX = x;
    intY = y;
    }

//Some public static data
public int intX, intY;
// Program entry point
public static int Main()
{
    //Trigger default constructor
    HelloClass c1 = new HelloClass();
    Console.WriteLine("c1.intX = {0}\n c1.intY = {1}\n",c1.intX, c1.intY);
    //Trigger Parameterized constructor…
    HelloClass c2;
    c2 = new HelloClass(100, 200);
    Console.WriteLine("c2.intX = {0}\n c2.intY = {1}\n",c2.intX, c2.intY);
    return 0;
}
}
```

20 A C# program to find the Square- root of a user entered number. The number is read from the keyboard.

```
using System;

namespace Console_App
{
    public class clsFactorial
    {
        public static void Main()
        {
            Console.WriteLine("Enter a number for Square Root:");
            int Number =Convert.ToInt32( Console.ReadLine());
            double SqrtNumber = Math.Sqrt(Number);
            Console.WriteLine("Square root of Number {0} is: {1}", Number, SqrtNumber);
            Console.ReadLine();
        }
    }
}
```

21 C# Program to find the smallest and largest of a given integer array set of elements.

```
using System;

namespace SamplePrograms
{
    class LargestSmallest
    {
        public static void Main()
```

```
        {
            // Declare and initialize the integer array
            int[] NumbersArray = { 1402, 834, 8289, 412, 1887, 29, 111};

            // Sort the array,
            Array.Sort(NumbersArray);

           //First element in the array will be smallest and the last element will be largest
            // Print the smallest number in the array
            Console.WriteLine("Samllest Number = {0}", NumbersArray[0]);

            // Print the largest number in the array.
            Console.WriteLine("Largest Number = {0}",
NumbersArray[NumbersArray.Length -1]);

            // Linq makes this much easier, as we have Min() and Max() extension methods
            // Console.WriteLine("Samllest Number = {0}", NumbersArray.Min());
            // Console.WriteLine("Largest Number = {0}", NumbersArray.Max());
        }
    }
}
```