**Dot Net Assignment**

**Dot Net MCQs:**

01. To prevent situations of runtime error during change or conversion of data type

02. Explicit Conversion

03. long, float, double

04. Common Language Runtime

05. 90

06. numbers are : 3 5 7 9 11

07. c += 1

08. &

**Dotnet Basic Questions:**

**Answer 1.**

Types of classes in C#: static classes, abstract classes, partial classes and sealed classes.

**Answer 2.**

Managed code and unmanaged code are terms used to describe two different types of code in the context of the .NET Framework.

Managed code refers to code that is executed by the Common Language Runtime (CLR), a component of the .NET Framework that manages memory, security, and other aspects of code execution. Managed code is written in languages such as C#, Visual Basic .NET, and F#

Unmanaged code, on the other hand, refers to code that is not executed by the CLR and is typically written in languages such as C++ or Assembly. Because unmanaged code does not run under the control of the CLR, it is responsible for managing its own memory and resources.

**Answer 3.**

In C#, extension methods are a language feature that allows you to add methods to existing classes or interfaces without modifying them directly. Extension methods are defined in a static class and are written as if they were instance methods of the type being extended.

**Answer 4.**

In C#, an Array and ArrayList are both used to store collections of objects, but they differ in their behavior and capabilities.

An array is a fixed-size collection of elements of the same type. Once an array is created, its size cannot be changed. Accessing an element in an array is very fast, as each element can be accessed directly by its index. But Arrays are fixed in size. An ArrayList, on the other hand, is a dynamic collection of objects that can grow or shrink in size as needed. Elements can be added or removed from an ArrayList at any time. Accessing an element in an ArrayList is slower than accessing an element in an array as each element must be retrieved sequentially.

**Answer 5.**

Boxing and unboxing are terms used in C# to describe the process of converting a value type to a reference type (boxing) and vice versa (unboxing).

**Answer 6.**

The Equality operator (==) compares the values of two variables, while the Equals() method compares the values of two objects. The Equals() method can be overridden to provide custom equality behavior.

**Answer 7.**

‘const’ is a compile-time constant that's evaluated at compile-time and can only be of certain types, while ‘readonly’ is a runtime constant that can be changed by the constructor and can be of any type. Use ‘const’ for values that are known at compile-time and are not expected to change, and use ‘readonly’ for values that are determined at runtime and cannot be modified after initialization.

**Answer 8.**

In C#, both String and StringBuilder are used to work with strings, but they differ in their behavior and usage.

String is an immutable type, which means that once a string is created, it cannot be changed. When you modify a string, what happens is that a new string is created with the modified value, and the old string is discarded.

StringBuilder, on the other hand, is a mutable type that's designed for efficient string manipulation. It allows you to modify a string without creating a new one each time.

**Dotnet Coding Problems:**

**Problem 01:**

using System;

using System.Text;

class Program

{

static void Main(string[] args)

{

Console.Write("Enter a string: ");

string input = Console.ReadLine();

StringBuilder sb = new StringBuilder();

for (int i = input.Length - 1; i >= 0; i--)

{

sb.Append(input[i]);

}

string reversed = sb.ToString();

Console.WriteLine($"Reversed string: {reversed}");

}

}

**Problem 02:**

using System;

using System.Linq;

class Program

{

static void Main(string[] args)

{

Console.Write("Enter a sentence: ");

string sentence = Console.ReadLine();

string[] words = sentence.Split(' ');

Array.Reverse(words);

string reversed = string.Join(" ", words);

Console.WriteLine($"Reversed sentence: {reversed}");

}

}

**Problem 03:**

using System;

class PalindromeCheck {

public static void Main(string[] args) {

string input = Console.ReadLine();

bool isPalindrome = true;

for (int i = 0; i < input.Length / 2; i++) {

if (input[i] != input[input.Length - 1 - i]) {

isPalindrome = false;

break; } }

if (isPalindrome)

Console.WriteLine("The string is a palindrome.");

else

Console.WriteLine("The string is not a palindrome.");

}

}

**Problem 04:**

using System;

class Program

{

static void Main(string[] args)

{

Console.Write("Enter a string: ");

string input = Console.ReadLine();

Console.Write("Enter the substring to find: ");

string substring = Console.ReadLine();

int index = input.IndexOf(substring);

if (index >= 0)

{

Console.WriteLine($"The substring '{substring}' was found at index {index}.");

}

else

{

Console.WriteLine($"The substring '{substring}' was not found in the input string.");

}

}

}