1. **Why do we use Async and Await in C#?**

Ans:-

1. It is basically used for asynchronous programming
2. Whenever there is large processing time require to process any kind of functionality (like reading/writing files, etc) the application has to wait to complete such execution in case of synchronous programming and that’s why we use Async await to achieve the asynchronous programming.
3. In Asynchronous programming we don’t have to wait for particular function to complete it’s execution, we can make it asyn and it call using it as await which states that current thread can move forward for execution.
4. Asyn function results the **Task** object which is an promising object which states that the where ever a result from that function required it will make it available for further execution.
5. **Using LINQ, reverse a string**

Ans:-

private static string ReverseStringUsingLinq(string message)

{

return new string(message.Reverse().ToArray());

}

1. **Given a date string ("dd-mm-yyyy") add 20 days to it and print it in** **yyyy-mm-dd format along with its day (Monday, Tuesday, ...)**

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Enter year in the format dd-mm-yyyy");

try

{

var dateString = Console.ReadLine()?.Split('-');

//By default it is trying to read the string in mm-dd-yyyy format

var convertedCorrectDateString = $"{dateString[1]}-{dateString[0]}-{dateString[2]}";

var date = Convert.ToDateTime(convertedCorrectDateString);

PrintTheDateInPredefinedFormat(date);

Console.ReadKey();

}

catch

{

Console.WriteLine("Invalid date format");

Console.ReadKey();

}

}

/// <summary>

/// Adds the specified number of days into date and print along with day

/// </summary>

/// <param name="date"></param>

/// <param name="days">no. of days to be added</param>

private static void PrintTheDateInPredefinedFormat(

DateTime date, int days=20)

{

var newDate = date.AddDays(days);

Console.WriteLine();

Console.WriteLine("Date after adding {0} days", days);

Console.WriteLine(newDate.ToString("yyyy-mm-dd") + " " + newDate.DayOfWeek);

}

}

1. **Write the necessary function here**

**void test() {**

**int a = 10;**

**int b = 20;**

**// call a function here**

**Console.Write($"a = {a} and b = {b} and c = {c}");**

**// which should print a = 20 and b = 10 and c = 200**

**// c is the product of a and b**

**}**

Ans:-

private static void Test()

{

int a = 10;

int b = 20;

// call a function here

int c = SwapAndMultiply(ref a, ref b);

Console.Write($"a = {a} and b = {b} and c = {c}");

// which should print a = 20 and b = 10 and c = 200

// c is the product of a and b

}

private static int SwapAndMultiply(ref int a, ref int b)

{

var temp = a;

a = b;

b = temp;

return a \* b;

}

1. **What can you do to make the following work**

**void func() {**

**string s = "some string";**

**if (s.isPalindrome()) {**

**Console.Write($"{s} is a palindrome")**

**}**

**}**

Ans:-

1. Add semi-colon at the end of Console.Write() statement;
2. Add extension method for palindrome as shown below

public static class Palindrom

{

public static bool IsPalindrome(this string message)

{

var reversedString = new string(message.Reverse().ToArray());

return reversedString == message;

}

}

1. **Build a user profile management application in .Net and send us the entire project. Make sure to add comments and to-dos cross your application to identify assumptions and highlight future recommended changes. The requirements for this simple “commandline” application is to allos the customer to:**
   * **Register a new user given their username, password, and email address**
   * **Verify a login using username and password**

**Please use unit tests to verify your critical functions with well documented test vectors. Send us the steps to compile and build this application. The target environment is Linux.**