Write a C# program that reverses a given string.

```
 public static void Main()
   {
     string myStr, rev;
    myStr = "Tom";
    rev ="";
    Int len = myStr.Length - 1;
    while (len >= 0) {
      rev = rev + myStr[len];
      len--;
    }
}
```

Original String: Hello, World!
Reversed String: !dlroW ,olleH

Fibonacci sequence

```
static void Main()
      {
         int n, a = 0, b = 1, c;
         Console.Write("Enter the number of terms : ");
         n = Int32.Parse(Console.ReadLine());
         for (int i = 0; i < n; i++)
         {
            if (i <= 1)
               c = i;
            else
            {
               c = a + b;
               a = b;
               b = c;
            }
            Console.Write(c+"  ");
         }
         Console.ReadKey();
      }
```

Fibonacci sequence with 10 terms:
0 1 1 2 3 5 8 13 21 34

string is a palindrome

```
using System.Linq;
return myString.SequenceEqual(myString.Reverse());

string rev = "";
for(int i=str.Length-1;i>=0;i--)
```

```
{
   rev += str[i].ToString();
}


 public static bool IsPalindrome(string str)
   {
      int left = 0;
      int right = str.Length - 1;

      while (left < right)
      {
         if (str[left] != str[right])
            return false;

         left++;
         right--;
      }
      return true;
   }
```
radar is a palindrome.

calculate the factorial of a given non-negative integer

```
   public static long CalculateFactorial(int n)
      {
         if (n < 0)
            throw new ArgumentException("Factorial is not defined for negative
numbers.");

    int result = 1;
   for(int i = n; i > 0; i--)
      result *= i;

         return result;
      }
```
Factorial of 5 is: 120 = 1*2*3*4*5

Swapping two numbers using a third variable in C#:

```
   int number1 = 10, number2 = 20, temp = 0;
   temp = number1; //temp=10
   number1 = number2; //number1=20
   number2 = temp; //number2=10
```

Swap two integers without using the third variable in C#

```
   int number1 = 10, number2 = 20;
   number1 = number1 * number2; //number1=200 (10*20)
   number2 = number1 / number2; //number2=10 (200/20)
```

```
number1 = number1 / number2; //number1=20 (200/10)
```

Given number is prime or not

```
public bool CalcIsPrime(int number) {

    if (number == 1) return false;
    if (number == 2) return true;

    if (number % 2 == 0) return false; // Even number
    for (int i = 2; i < number; i++)
   { // Advance from two to include correct calculation
        for '4'
      if (number % i == 0) return false;
    }
    return true;
}
```

Prime number is a number that can be divided either by itself or 1

Armstrong Number Program in C#

An Armstrong Number is a number that is equal to the sum of, power of each digit by the total number of digits.

```
371 = (3*3*3)+(7*7*7)+(1*1*1)
where:
(3*3*3)=27
(7*7*7)=343
(1*1*1)=1
So: 27+343+1=371
int  n,r,sum=0,temp;
      Console.Write("Enter the Number= ");
      n= int.Parse(Console.ReadLine());
      temp=n;
      while(n>0)
      {
       r=n%10;
       sum=sum+(r*r*r);
       n=n/10;
      }
      if(temp==sum)
       Console.Write("Armstrong Number.");
      else
       Console.Write("Not Armstrong Number.");
     }
```

Sum of Digits Program in C# with Examples:

```
int number = int.Parse(Console.ReadLine());
          int sum = 0, reminder;

          while (number > 0)
```

```
        {
            reminder = number % 10;
            sum = sum + reminder;
            number = number / 10;
        }
```

Remove Duplicate Characters from a String in C#

```
string inputString = Console.ReadLine();
      string resultString = string.Empty;
      for (int i = 0; i < inputString.Length; i++)
      {
         if (!resultString.Contains(inputString[i]))
         {
            resultString += inputString[i];
         }
      }

var uniqueCharArray = inputString.ToCharArray().Distinct().ToArray();
var resultString = new string(uniqueCharArray);
```

Find All Substrings of a Given String in C#

```
for (int i = 0; i < inputString.Length; ++i)
        {
            StringBuilder subString = new StringBuilder(inputString.Length - i);
            for (int j = i; j < inputString.Length; ++j)
            {
               subString.Append(inputString[j]);
               Console.Write(subString + " ");
            }
        }

//This loop maintains the starting character
        for (int i = 0; i < len; i++)
        {
            //This loop adds the next character every iteration for the substring and
then print
            for (int j = 0; j < len - i; j++)
            {
               Console.Write (inputString.Substring(i, j + 1) + " ");
            }
        }
```

Remove Duplicate Elements from an Array in C#

```
   Print number occure only once
   int i = 0, j = 0;
     int[] arr1 = new int[]{ 7, 7, 8, 8, 9, 1, 1, 4, 2, 2 };
```

```
        for (i = 0; i < arr1.Length; i++)
        {
         for (j = 0; j < arr1.Length; j++)
         {
            if (i == j)
            continue;
            if (arr1[j] == arr1[i])
            break;
         }
         if (arr1.Length == j)
         {
            Console.Write (arr1[i] + " ");
         }
        }

  //
  int[] s = { 1, 2, 3, 3, 4};
  int[] q = s.Distinct().ToArray();

  //
  string[] sArray = {"a", "b", "b", "c", "c", "d", "e", "f", "f"};
  var sList = new ArrayList();

  for (int i = 0; i < sArray.Length; i++) {
     if (sList.Contains(sArray[i]) == false) {
        sList.Add(sArray[i]);
     }
  }
```

Strong Number Program in C#

A Strong number is a special number whose sum of the factorial of each digit is equal to the original number

Input: 145

Explanation: 1! + 4! + 5! = 145

1 + (1*2*3*4) + (1*2*3*4*5) = 145

```
// Function to calculate the factorial of a number
 static int Factorial(int num) {
  return (num == 0 || num == 1) ? 1 : num * Factorial(num - 1);
 }

// Function to check if a number is a Strong Number
 static bool IsStrongNumber(int num) {
  int originalNum = num;
  int sum = 0;

  while (num > 0) {
```

```
    int digit = num % 10;
    sum += Factorial(digit);
    num /= 10;
  }

  return (sum == originalNum);
}
```