**C# Lab Programs**

1.Develop a C# program to simulate simple airthmetic calculator for addition, subtraction, multiplication, division and mod opertaions. Read the operator and operands through console.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace C\_\_Program\_1

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("enter the number1");

float number1 = Convert.ToSingle(Console.ReadLine());

Console.WriteLine("enter the number2");

float number2 = Convert.ToSingle(Console.ReadLine());

Console.WriteLine("enter the operator");

char operation = char.Parse(Console.ReadLine());

double result = 0;

switch (operation)

{

case '+':

result = number1 + number2;

break;

case '-':

result = number1 - number2;

break;

case '\*':

result = number1 \* number2;

break;

case '/':

if (number2 != 0)

{

result = number1 / number2;

}

else

{

Console.WriteLine("division by zero is not allowed");

Console.ReadLine();

}

break;

case '%':

if (number2 != 0)

{

result = number1 % number2;

}

else

{

Console.WriteLine("Modulus by zero is not allowed");

Console.ReadLine();

}

break;

default:

Console.WriteLine("invalid operator");

Console.ReadLine();

return;

}

Console.WriteLine("Result:" + number1 + " " + operation + " " + number2 + " = " + result);

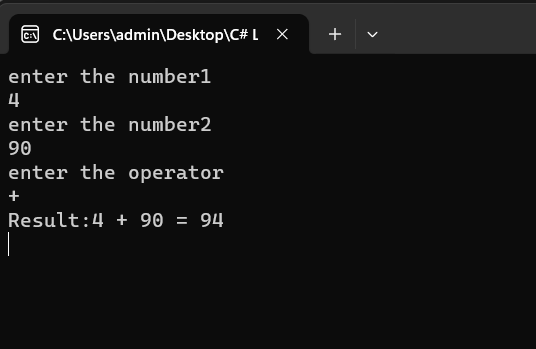
Console.ReadLine();

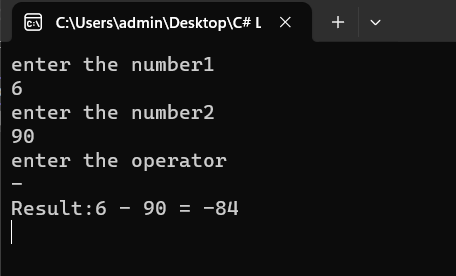
}

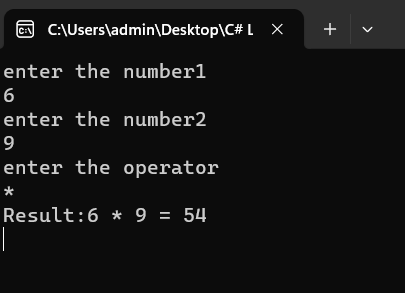
}

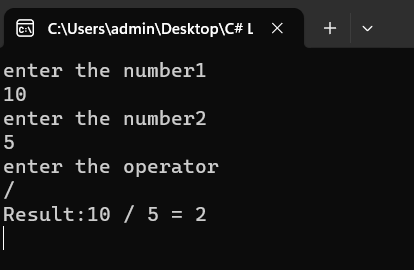
}

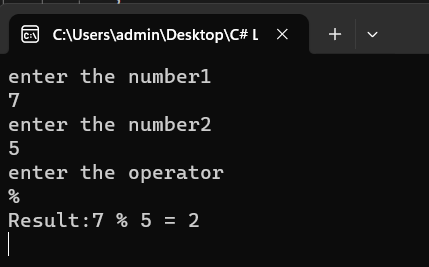
**OUTPUT**

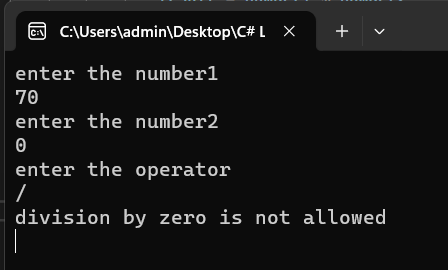


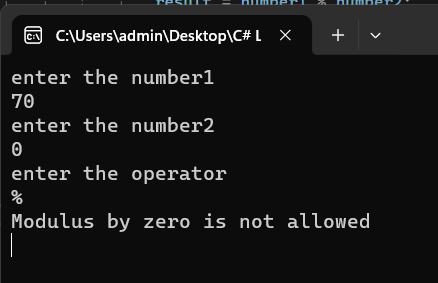












2. Develop a C# program to print Armstrong number between 1 to 1000.

using ArmstrongNumber;

using System;

namespace ArmstrongNumber

{

class Program

{

public void Check\_ArmstrongNumber(int number)

{

int originalNumber = number;

int n = CountDigit(number);

int sum = 0;

while (number > 0)

{

int digit = number % 10;

sum = sum + (int)Math.Pow(digit, n);

number = number / 10;

}

if (sum == originalNumber)

{

Console.WriteLine(originalNumber);

}

}

public int CountDigit(int number)

{

int count = 0;

while (number > 0)

{

count++;

number = number / 10;

}

return count;

}

}

class MainClass

{

static void Main(String[] args)

{

Program obj = new Program();

Console.WriteLine("Armstrong number between 1 to 1000 are");

for (int i = 1; i <= 1000; i++)

{

obj.Check\_ArmstrongNumber(i);

}

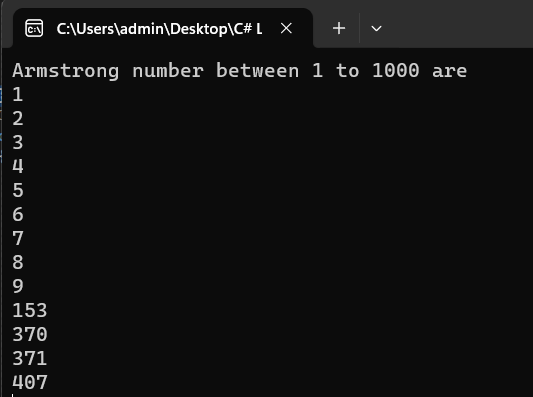
Console.ReadLine();

}

}

}

OUTPUT



3. Develop a c# program to list all substring in a given string[Hint:use of Substring() method].

using System;

namespace Substrings

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("enter the string");

string str=Console.ReadLine();

Console.WriteLine("the possible substrings in the given string are");

Getsubstring(str);

}

static void Getsubstring(string str)

{

for(int i=0; i<str.Length; i++)

{

for(int j=i+1; j<=str.Length; j++)

{

string substring=str.Substring(i,j-i);

Console.WriteLine(substring);

}

}

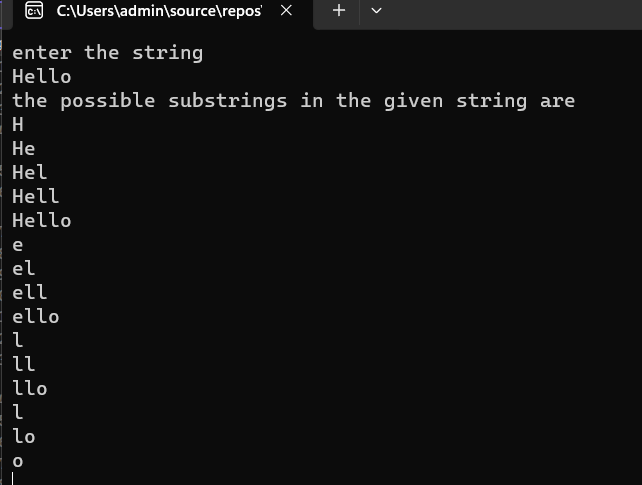
Console.ReadLine();

}

}

}

OUTPUT



4. Develop a C# program to demonstrate Diision by Zero and Index Out of Range exception.

using System;

namespace ExceptionHandling

{

class Program

{

public static void Main(string[] args)

{

Console.WriteLine("Enter numerator:");

float numerator = int.Parse(Console.ReadLine());

Console.WriteLine("Enter denominator:");

float denominator = int.Parse(Console.ReadLine());

try

{

if (denominator == 0)

{

throw new DivideByZeroException("Cannot divide by zero.");

}

float result = numerator / denominator;

Console.WriteLine("Result of division:"+result);

}

catch (DivideByZeroException ex)

{

Console.WriteLine("Divide by zero exception:"+ex.Message);

}

Console.WriteLine("Enter the array size:");

int arraySize = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter the array elements:");

try

{

int[] array = new int[arraySize]; //creating an array

for (int i = 0; i < arraySize; i++)

{

array[i] = Convert.ToInt32(Console.ReadLine());

}

Console.WriteLine("enter the array position to access value");

int index = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("value in the givin position is:"+array[index]);

}

catch (IndexOutOfRangeException ex)

{

Console.WriteLine("Index out of range exception:" +ex.Message);

}

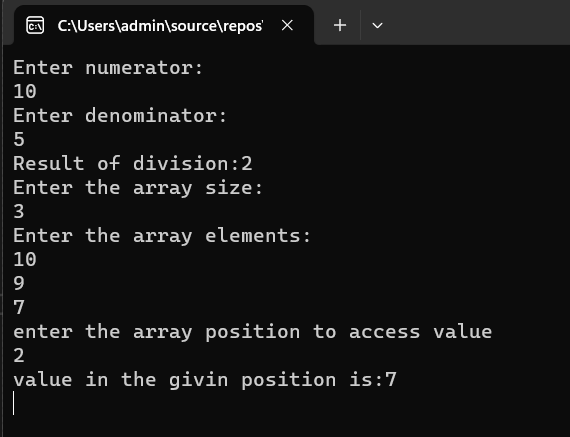
Console.ReadLine();

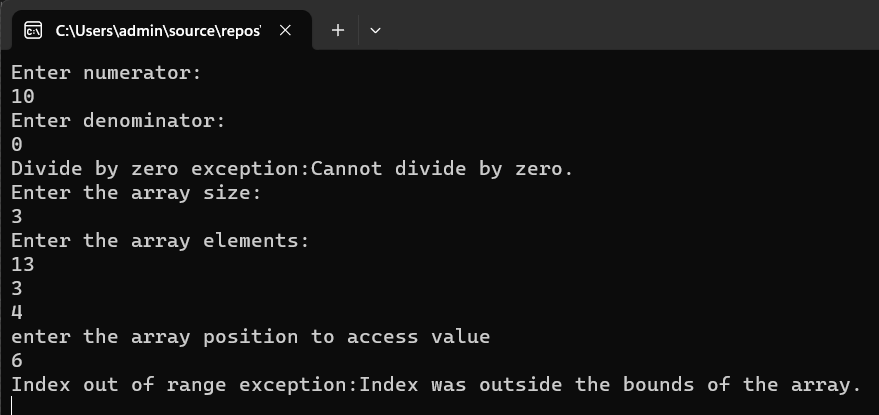
}

}

}

OUTPUT





**5. Develop a C# Program to Generate and Print Pascal Triangle using Two Dimensional Arrays.**

using System;

class Program

{

public static void Main(String[] args)

{

Console.WriteLine("Enter the number of rows");

int Rows = int.Parse(Console.ReadLine());

int[,] pascalTriangle = Generate(Rows);

Print(pascalTriangle);

Console.ReadLine();

}

static int[,] Generate(int Rows)

{

int[,] triangle = new int[Rows, Rows];

for (int i = 0; i < Rows; i++)

{

triangle[i, 0] = 1;

for (int j = 1; j < i; j++)

{

triangle[i, j] = triangle[i - 1, j - 1] + triangle[i - 1, j];

}

triangle[i, i] = 1;

}

return triangle;

}

static void Print(int[,] triangle)

{

Console.WriteLine("Pascal's Triangle:");

for (int i = 0; i < triangle.GetLength(0); i++)

{

// Add leading spaces for formatting

for (int space = 0; space < triangle.GetLength(0) - i - 1; space++)

{

Console.Write(" ");

}

for (int j = 0; j <= i; j++)

{

Console.Write(triangle[i, j] + " ");

}

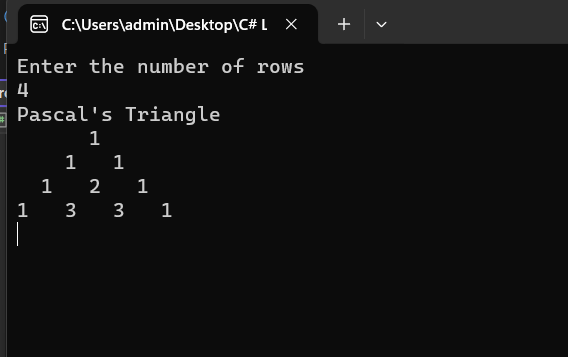
Console.WriteLine();

}

}

}

**OUTPUT**

****

**6. Develop a C# Program to Generate and Print Floyds Triangle using Jagged arrays.**

using System;

class Program

{

public static void Main(String[] args)

{

Console.WriteLine("Enter the number of rows");

int Rows = int.Parse(Console.ReadLine());

Console.WriteLine("Floyd’s Triangle");

Print(Rows);

Console.ReadLine();

}

static void Print(int Rows)

{

int[][] triangle = new int[Rows][];

int value = 1;

for (int i = 0; i < Rows; i++)

{

triangle[i] = new int[i + 1];

for (int j = 0; j <= i; j++)

{

Console.Write(triangle[i][j]+ value++ +" ");

}

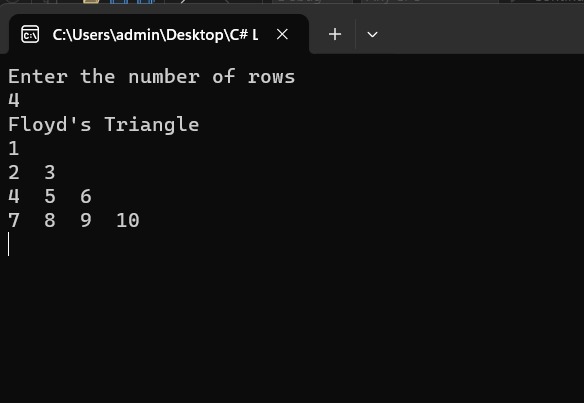
Console.WriteLine();

}

}

}

**OUTPUT**

****