Contents

[Assignments Discussion with extra program 2](#_Toc164615730)

[1. Read in a non-negative integer (print out 'crazy input' if the input is not given correctly) and check if it is a Curzon Number. A Curzon number is a number N where 2^N + 1 is divisible by 2\*N + 1. 2](#_Toc164615731)

[2. Find all roots of a quadratic equation. The standard form of a quadratic equation is: ax² + bx + c = 0 4](#_Toc164615732)

[3. Read in a character and check whether a character is a vowel, a consonant, a digit or a special character. 5](#_Toc164615733)

[4. Read in three sides of a triangle and print out whether the triangle is equilateral (all equal sides), isosceles (equal legs, so only 2 equal sides) or scalene (no equal sides). Sides can only be positive... otherwise, it's just 'crazy input' 6](#_Toc164615734)

[5. Read in a value that will determine the depth of a pond. If the pond is 4 deep, the bottom will be at that 4th level. 7](#_Toc164615735)

[Tip: try to look for patterns and don't be afraid to split this up into sections. 7](#_Toc164615736)

[6.Read the n-th position of a Fibonacci number of which you have to find the previous Fibonacci number and the current one. Use n to determine the position in the Fibonacci sequence. 9](#_Toc164615737)

[7. *Prime numbers:* Read in a number n and print all prime numbers between 2 and n. (when n < 2 print out 'crazy input'). A prime number is an integer larger than 1, which has only 2 divisors: 1 and itself. 10](#_Toc164615738)

[8. Read in a roman numeral n and convert it to a decimal number. (M = 1000, D = 500, C = 100, L = 50, X = 10, V = 5 and I = 1) 12](#_Toc164615739)

[Example: n = DXII --> 500 + 10 + 1 + 1 = 512 12](#_Toc164615740)

[9. Read in a string and count the vowels, consonants, digits, and other characters (exclude spaces!). 13](#_Toc164615741)

[10. Read in a string and check if it is an isogram. 15](#_Toc164615742)

[11. Print all the armstrong numbers between 1 to 1000 17](#_Toc164615743)

[12. ATM Machine Transactions Program in C# 19](#_Toc164615744)

[13. Read in an array of characters. (Your're allowed to use .ToCharArray()) 22](#_Toc164615745)

[Count for each character in the array the frequency of it's occurence. 22](#_Toc164615746)

[14. Read in two strings (separated by a space) and find the longest common ending between two given strings. 24](#_Toc164615747)

[15 Read in an array of integers split by space. Check whether or not each number is even or odd. 26](#_Toc164615748)

[Output these results into an array in which Even or Odd take the place of the number it corresponds to. 26](#_Toc164615749)

[16. Read in a sorted array and a value. Insert the value in the right place and print out the new array. 27](#_Toc164615750)

[17. Read in an array of integers and find the missing number in that array. 29](#_Toc164615751)

[18. Read a string and transform it to an array. Take each word from that array and reverse the words. Print out the new array. 31](#_Toc164615752)

[19. Read in an array of integers and rotate it a specified number (never greater than array length!) of places to the left. 32](#_Toc164615753)

[20. Read in an array of numbers (separated by a space), and find the second largest element. 35](#_Toc164615754)

[21. Read in an intiger, indicating a square array. Next, read in every element of this 2D-array string by string, split with a space. 36](#_Toc164615755)

[Lastly, swap around the rows and columns and print out your new array. 36](#_Toc164615756)

# Assignments Discussion with extra program

## Read in a non-negative integer (print out 'crazy input' if the input is not given correctly) and check if it is a Curzon Number. A Curzon number is a number N where 2^N + 1 is divisible by 2\*N + 1.

using System;

class Program

{

static void Main()

{

Console.WriteLine("Enter a non-negative integer:");

string input = Console.ReadLine();

if (!int.TryParse(input, out int n) || n < 0)

{

Console.WriteLine("Crazy input!");

}

else

{

long numerator = (long)Math.Pow(2, n) + 1;

long denominator = 2 \* n + 1;

bool isCurzon = numerator % denominator == 0;

Console.WriteLine($"Is Curzon Number: {isCurzon}");

}

}

}

**About** : **int.TryParse(input, out int n)**   
  
The **int.TryParse** method in C# is used to convert a string representation of a number to its integer equivalent. It's a safer alternative to **int.Parse** because it doesn't throw an exception if the conversion fails.

Here's how it works:

1. **int.TryParse(input, out int n)** takes two parameters:
   * **input**: The string that you want to convert to an integer.
   * **out int n**: The variable where the converted integer will be stored if the conversion is successful.
2. **int.TryParse** returns a boolean value:
   * If the conversion is successful, it returns **true**, and the converted integer is stored in the variable **n**.
   * If the conversion fails (for example, if **input** is not a valid integer), it returns **false**, and **n** will be set to **0**.

## Find all roots of a quadratic equation. The standard form of a quadratic equation is: ax² + bx + c = 0

using System;

class Program

{

static void Main()

{

Console.WriteLine("Enter the coefficients of the quadratic equation (a, b, c):");

string[] coefficients = Console.ReadLine().Split(' ');

if (coefficients.Length != 3)

{

Console.WriteLine("Invalid input. Please enter three coefficients.");

return;

}

if (!double.TryParse(coefficients[0], out double a) ||

!double.TryParse(coefficients[1], out double b) ||

!double.TryParse(coefficients[2], out double c))

{

Console.WriteLine("Invalid input. Coefficients must be numeric.");

return;

}

double discriminant = b \* b - 4 \* a \* c;

if (discriminant < 0)

{

Console.WriteLine("The equation has no real roots.");

return;

}

double root1 = (-b + Math.Sqrt(discriminant)) / (2 \* a);

double root2 = (-b - Math.Sqrt(discriminant)) / (2 \* a);

Console.WriteLine($"Root 1: {root1}");

Console.WriteLine($"Root 2: {root2}");

}

}

.Split(' '): This method splits the input line into an array of strings based on the space character (' '). For example, if the user enters "2 3 4", this method will create an array ["2", "3", "4"].

## Read in a character and check whether a character is a vowel, a consonant, a digit or a special character.

using System;

class Program

{

static void Main()

{

Console.WriteLine("Enter a character:");

char input = Console.ReadKey().KeyChar;

if ((input >= 'a' && input <= 'z') || (input >= 'A' && input <= 'Z'))

{

if (input == 'a' || input == 'e' || input == 'i' || input == 'o' || input == 'u' ||

input == 'A' || input == 'E' || input == 'I' || input == 'O' || input == 'U')

{

Console.WriteLine("Vowel");

}

else

{

Console.WriteLine("Consonant");

}

}

else if (input >= '0' && input <= '9')

{

Console.WriteLine("Digit");

}

else

{

Console.WriteLine("Special Character");

}

}

}

## Read in three sides of a triangle and print out whether the triangle is equilateral (all equal sides), isosceles (equal legs, so only 2 equal sides) or scalene (no equal sides). Sides can only be positive... otherwise, it's just 'crazy input'

using System;

class Program

{

static void Main()

{

Console.WriteLine("Enter the three sides of the triangle (positive numbers only):");

double x = double.Parse(Console.ReadLine());

double y = double.Parse(Console.ReadLine());

double z = double.Parse(Console.ReadLine());

if (x <= 0 || y <= 0 || z <= 0)

{

Console.WriteLine("Crazy input! Please enter positive numbers only.");

return;

}

if (x == y && y == z)

{

Console.WriteLine("Equilateral");

}

else if (x == y || x == z || y == z)

{

Console.WriteLine("Isosceles");

}

else

{

Console.WriteLine("Scalene");

}

}

}

## 5. Read in a value that will determine the depth of a pond. If the pond is 4 deep, the bottom will be at that 4th level.

## Tip: try to look for patterns and don't be afraid to split this up into sections.

Input/Output

input:

5

output:

0 ~ ~ ~ ~ ~ ~ ~ ~ 0

0 0 ~ ~ ~ ~ ~ ~ 0 0

0 0 0 ~ ~ ~ ~ 0 0 0

0 0 0 0 ~ ~ 0 0 0 0

0 0 0 0 0 0 0 0 0 0

input:

3

output:

0 ~ ~ ~ ~ 0

0 0 ~ ~ 0 0

0 0 0 0 0 0

using System;

class Program

{

static void Main()

{

// Prompt the user to enter the depth of the pond.

Console.WriteLine("Enter the depth of the pond:");

// Read the depth of the pond from the user.

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

// Loop through each row of the pond.

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

{

// Print the left side of the pond.

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

{

Console.Write("0 ");

}

// Print the water in the pond.

for (int k = depth - 1; k > i; k--)

{

Console.Write("~ ");

}

// Print the water in the pond.

for (int l = depth - 1; l > i; l--)

{

Console.Write("~ ");

}

// Print the right side of the pond.

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

{

Console.Write("0 ");

}

// Move to the next line.

Console.WriteLine();

}

}

}

## 6.Read the n-th position of a Fibonacci number of which you have to find the previous Fibonacci number and the current one. Use n to determine the position in the Fibonacci sequence.

using System;

class Program

{

static void Main()

{

Console.WriteLine("Enter the position of the Fibonacci number:");

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

int previous = 0;

int current = 1;

for (int i = 2; i <= n; i++)

{

int next = previous + current;

previous = current;

current = next;

}

Console.WriteLine($"The previous Fibonacci number is: {previous}");

Console.WriteLine($"The current Fibonacci number is: {current}");

}

}

## 7. *Prime numbers:* Read in a number n and print all prime numbers between 2 and n. (when n < 2 print out 'crazy input'). A prime number is an integer larger than 1, which has only 2 divisors: 1 and itself.

using System;

class Program

{

static void Main()

{

Console.WriteLine("Enter a number:");

if (int.TryParse(Console.ReadLine(), out int num))

{

if (num < 2)

{

Console.WriteLine("Crazy input");

}

else

{

Console.WriteLine("Prime numbers between 2 and " + num + ":");

// Loop through each number from 2 to num.

for (int i = 2; i <= num; i++)

{

bool isPrime = true;

// Check if the current number is prime.

for (int j = 2; j \* j <= i; j++)

{

if (i % j == 0)

{

isPrime = false;

break;

}

}

// Print the prime number.

if (isPrime)

{

Console.Write(i + " ");

}

}

}

}

else

{

Console.WriteLine("Invalid input");

}

}

}

## 8. Read in a roman numeral n and convert it to a decimal number. (M = 1000, D = 500, C = 100, L = 50, X = 10, V = 5 and I = 1)

## Example: n = DXII --> 500 + 10 + 1 + 1 = 512

using System;

namespace \_04Roman

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("Enter a Roman numeral:");

string roman = Console.ReadLine().ToUpper();

int decimalnum = 0;

int previous = 0;

foreach (char numeral in roman)

{

int value = 0;

switch (numeral)

{

case 'I':

value = 1;

break;

case 'V':

value = 5;

break;

case 'X':

value = 10;

break;

case 'L':

value = 50;

break;

case 'C':

value = 100;

break;

case 'D':

value = 500;

break;

case 'M':

value = 1000;

break;

default:

Console.WriteLine("Invalid input: " + numeral);

return;

}

decimalnum += value;

if (previous < value)

decimalnum -= 2 \* previous;

previous = value;

}

Console.Write(decimalnum);

}

}

}

## 9. Read in a string and count the vowels, consonants, digits, and other characters (exclude spaces!).

using System;

class Program

{

static void Main()

{

// Prompt the user to enter a string.

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

// Read the input as a string.

string input = Console.ReadLine();

// Initialize counters for vowels, consonants, digits, and other characters.

int vowels = 0;

int consonants = 0;

int digits = 0;

int others = 0;

// Loop through each character in the input string.

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

{

// Get the current character.

char ch = input[i];

// Check if the character is a vowel.

if ("AEIOUaeiou".IndexOf(ch) >= 0)

{

vowels++;

}

// Check if the character is a consonant.

else if (char.IsLetter(ch))

{

consonants++;

}

// Check if the character is a digit.

else if (char.IsDigit(ch))

{

digits++;

}

// Otherwise, the character is an other character.

else if (ch != ' ')

{

others++;

}

}

// Print the counts of vowels, consonants, digits, and other characters.

Console.WriteLine("Vowels: " + vowels);

Console.WriteLine("Consonants: " + consonants);

Console.WriteLine("Digits: " + digits);

Console.WriteLine("Other characters: " + others);

}

}

Note :

The IndexOf(ch) method in C# is used to find the position (index) of a specific character ch within a string. It returns the first occurrence of ch in the string. If ch is not found in the string, it returns -1.

## 10. Read in a string and check if it is an isogram.

using System;

namespace \_05Isogram

{

internal class Program

{

static void Main(string[] args)

{

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

string input = Console.ReadLine();

string lowercaseString = input.ToLower();

bool isIsogram = true;

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

{

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

{

if (lowercaseString[i] == lowercaseString[j] && lowercaseString[i] != ' ')

{

isIsogram = false;

break;

}

}

if (!isIsogram) break;

}

if (isIsogram)

{

Console.Write("True");

}

else

{

Console.Write("False");

}

}

}

}

## 11. Print all the armstrong numbers between 1 to 1000

using System;

class MainClass

{

public static void Main(string[] args)

{

Console.WriteLine("Armstrong numbers between 1 and 1000:");

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

{

int sum = 0;

int temp = num;

int numDigits = num.ToString().Length;

while (temp != 0)

{

int digit = temp % 10;

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

temp /= 10;

}

if (num == sum)

{

Console.WriteLine(num);

}

}

}

}

**Armstrong number between specific range :**

using System;

class MainClass

{

public static void Main(string[] args)

{

Console.WriteLine("Enter the range:");

Console.Write("From: ");

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

Console.Write("To: ");

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

Console.WriteLine($"Armstrong numbers between {from} and {to}:");

for (int num = from; num <= to; num++)

{

int sum = 0;

int temp = num;

int numDigits = num.ToString().Length;

while (temp != 0)

{

int digit = temp % 10;

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

temp /= 10;

}

if (num == sum)

{

Console.WriteLine(num);

}

}

}

}

## 12. ATM Machine Transactions Program in C#

using System;

namespace atm\_program {

class ab {

static void Main(String[] args) {

int amt = 10000, a, current, pin = 4040, pin1, pin2;

//read PIN

Console.WriteLine("Enter the pin");

pin1 = int.Parse(Console.ReadLine());

//compare PIN

if (pin1 == pin) {

Console.WriteLine("1.To check balance");

Console.WriteLine("2.To withdraw money");

Console.WriteLine("3.To deposite Money");

Console.WriteLine("4.To change the pin");

Console.WriteLine("Enter your choice");

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

switch (ch) {

case 1:

Console.WriteLine("The current balance in your account is" + amt);

break;

case 2:

Console.WriteLine("Enter the amount to withdraw"); {

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

if (amt >= a) {

if (a % 100 == 0) {

Console.WriteLine("Please collect the cash" + a);

current = amt - a;

Console.WriteLine("The current balance is now" + current);

} else

Console.WriteLine("Please enter the amount to withdraw in the multiples of 100");

} else

Console.WriteLine("Your account does not have sufficient balance");

}

break;

case 3:

Console.WriteLine("Enter the amount to be deposite");

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

current = amt + a;

Console.WriteLine("The current balance in the account is" + current);

break;

case 4:

Console.WriteLine("Want to change your pin");

Console.WriteLine("Enter your previous pin");

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

if (prepin == pin) {

Console.WriteLine("Enter your new pin");

pin2 = int.Parse(Console.ReadLine());

pin1 = pin2;

Console.WriteLine("Your pin is changed");

} else

Console.WriteLine("Enter your correct pin");

break;

default:

Console.WriteLine("Please select correct option");

break;

}

} else

Console.WriteLine("Pin is wrong");

}

}

}

## 13. Read in an array of characters. (Your're allowed to use .ToCharArray())

## Count for each character in the array the frequency of it's occurence.

Input/Output

input:

hello, world!

output:

Character ' ': 1 times

Character '!': 1 times

Character ',': 1 times

Character 'd': 1 times

Character 'e': 1 times

Character 'h': 1 times

Character 'l': 3 times

Character 'o': 2 times

Character 'r': 1 times

Character 'w': 1 times

using System;

class Program

{

static void Main()

{

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

string input = Console.ReadLine();

int[] charFrequency = new int[256]; // Array to store frequency of each character (ASCII values)

// Convert input string to character array

char[] charArray = input.ToCharArray();

// Count the frequency of each character

foreach (char c in charArray)

{

int asciiValue = (int)c;

charFrequency[asciiValue]++;

}

// Display character frequencies

Console.WriteLine("\nCharacter frequencies:");

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

{

if (charFrequency[i] > 0)

{

char character = (char)i;

Console.WriteLine("Character '" + character + "': " + charFrequency[i] + " times");

}

}

}

}

## 14. Read in two strings (separated by a space) and find the longest common ending between two given strings.

Note: use of Linq-library or Array-library is forbidden!

Input/Output

input:

dream stream

output:

ream

using System;

class Program

{

static void Main()

{

Console.WriteLine("Enter two strings separated by a space:");

string input = Console.ReadLine();

// Split the input into two strings

string[] parts = input.Split(' ');

string firstString = parts[0];

string secondString = parts[1];

// Find the longest common ending

int len1 = firstString.Length;

int len2 = secondString.Length;

int index1 = len1 - 1;

int index2 = len2 - 1;

// Find the longest common ending by comparing characters from the end

while (index1 >= 0 && index2 >= 0 && firstString[index1] == secondString[index2])

{

index1--;

index2--;

}

// Extract and print the longest common ending

string longestEnding = firstString.Substring(index1 + 1);

Console.WriteLine("Longest common ending: " + longestEnding);

}

}

The Substring method in C# is used to extract a substring from a given string. It takes one or two parameters:

Start Index: The index at which to begin the substring.

Length (Optional): The length of the substring to extract. If omitted, the substring extends to the end of the original string.

Syntax:  
string Substring(int startIndex);

string Substring(int startIndex, int length);

## 15 Read in an array of integers split by space. Check whether or not each number is even or odd.

## Output these results into an array in which Even or Odd take the place of the number it corresponds to.

Input/Output

input:

5 -2 3 18 99 27 4 16

output:

Odd Even Odd Even Odd Odd Even Even

using System;

class Program

{

static void Main()

{

Console.WriteLine("Enter integers separated by space:");

string input = Console.ReadLine();

// Split the input string into individual numbers

string[] numbersStr = input.Split(' ');

// Create an array to store even/odd results

string[] results = new string[numbersStr.Length];

// Check each number and store the result

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

{

int num = int.Parse(numbersStr[i]);

results[i] = (num % 2 == 0) ? "Even" : "Odd";

}

// Print the results

Console.WriteLine("Output:");

Console.WriteLine(string.Join(" ", results));

}

}

## 16. Read in a sorted array and a value. Insert the value in the right place and print out the new array.

Input/Output

input:

-9 -4 -1 2 13 25 83

-7

output:

-9 -7 -4 -1 2 13 25 83

using System;

namespace \_06Insert

{

internal class Program

{

static void Main(string[] args)

{

Console.WriteLine("Enter a sorted array separated by spaces:");

string inputArrayStr = Console.ReadLine();

Console.WriteLine("Enter the value to insert:");

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

string[] inputArray = inputArrayStr.Split(' ');

int[] sortedArray = new int[inputArray.Length + 1];

int i = 0;

bool inserted = false;

foreach (string element in inputArray)

{

int number = int.Parse(element);

if (!inserted && number > valueToInsert)

{

sortedArray[i] = valueToInsert;

inserted = true;

i++;

}

sortedArray[i] = number;

i++;

}

if (!inserted)

{

sortedArray[i] = valueToInsert;

}

Console.WriteLine("Output:");

foreach (int number in sortedArray)

{

Console.Write(number + " ");

}

}

}

}

## 17. Read in an array of integers and find the missing number in that array.

input:

7 2 9 1 6 5 8 3

output:

4

using System;

class Program

{

static void Main()

{

Console.WriteLine("Enter integers separated by space:");

string input = Console.ReadLine();

// Split the input string into individual numbers

string[] numbersStr = input.Split(' ');

int[] numbers = new int[numbersStr.Length];

// Convert the string array to an integer array

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

{

numbers[i] = int.Parse(numbersStr[i]);

}

// Sort the array to easily find the missing number

Array.Sort(numbers);

// Find the missing number

int n = numbers.Length + 1;

int totalSum = (n \* (n + 1)) / 2;

int actualSum = 0;

foreach (int num in numbers)

{

actualSum += num;

}

int missingNumber = totalSum - actualSum == 0 ? -1 : totalSum - actualSum;

// Print the result

if (missingNumber != -1)

{

Console.WriteLine("Missing number: " + missingNumber);

}

else

{

Console.WriteLine("There is no missing number!");

}

}

}

## 18. Read a string and transform it to an array. Take each word from that array and reverse the words. Print out the new array.

Input/Output

input:

Once I make my move, the Queen will take me. Then you're free to check the king.

output:

ecnO I ekam ym ,evom eht neeuQ lliw ekat .em nehT er'uoy eerf ot kcehc eht .gnik

using System;

namespace \_06Reverse

{

internal class Program

{

static void Main(string[] args)

{

// Read input from the console

string input = Console.ReadLine();

// Split the input string into an array of words

string[] separators = { " ", "\t", "\n", "\r" };

string[] words = input.Split(separators, StringSplitOptions.RemoveEmptyEntries);

// Reverse each word in the array

foreach (string word in words)

{

// Reverse a word manually

char[] charArray = new char[word.Length];

int j = 0;

for (int k = word.Length - 1; k >= 0; k--)

{

charArray[j++] = word[k];

}

string reversedWord = new string(charArray);

Console.Write(reversedWord + " ");

}

}

}

}

## 19. Read in an array of integers and rotate it a specified number (never greater than array length!) of places to the left.

input:

9 8 7 6 5 4 3 2 1

5

output:

4 3 2 1 9 8 7 6 5

using System;

class Program

{

static void Main()

{

// Read input array from the console

Console.WriteLine("Enter integers separated by space:");

string input = Console.ReadLine();

string[] numbersStr = input.Split(' ');

int[] numbers = new int[numbersStr.Length];

// Convert string array to integer array

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

{

numbers[i] = int.Parse(numbersStr[i]);

}

// Read the number of places to rotate

Console.WriteLine("Enter the number of places to rotate:");

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

// Perform rotation

int n = numbers.Length;

rotation %= n; // Ensure places is less than array length

// Reverse the first part of the array (0 to rotation-1)

for (int i = 0; i < rotation / 2; i++)

{

int temp = numbers[i];

numbers[i] = numbers[rotation - 1 - i];

numbers[rotation - 1 - i] = temp;

}

// Reverse the second part of the array (rotation to n-1)

for (int i = rotation, j = n - 1; i < j; i++, j--)

{

int temp = numbers[i];

numbers[i] = numbers[j];

numbers[j] = temp;

}

// Reverse the entire array

for (int i = 0, j = n - 1; i < j; i++, j--)

{

int temp = numbers[i];

numbers[i] = numbers[j];

numbers[j] = temp;

}

// Print the rotated array

Console.WriteLine("Output:");

foreach (int num in numbers)

{

Console.Write(num + " ");

}

}

}

## 20. Read in an array of numbers (separated by a space), and find the second largest element.

using System;

class Program

{

static void Main()

{

// Step 1: Read the Input

Console.WriteLine("Enter numbers separated by spaces:");

string input = Console.ReadLine();

// Step 2: Convert to Array

string[] numbersStr = input.Split(' ');

int[] numbers = new int[numbersStr.Length];

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

{

numbers[i] = int.Parse(numbersStr[i]);

}

// Step 3: Find the Largest Number

int largest = int.MinValue;

foreach (int num in numbers)

{

if (num > largest)

{

largest = num;

}

}

// Step 4: Find the Second Largest Number

int secondLargest = int.MinValue;

foreach (int num in numbers)

{

if (num != largest && num > secondLargest)

{

secondLargest = num;

}

}

// Step 5: Output the Result

Console.WriteLine("Second largest element: " + secondLargest);

}

}

## 21. Read in an intiger, indicating a square array. Next, read in every element of this 2D-array string by string, split with a space.

## Lastly, swap around the rows and columns and print out your new array.

(same as transpose of matrix)

input:

4

9 7 5 3

1 2 3 4

5 6 7 8

1 9 1 9

output:

9 1 5 1

7 2 6 9

5 3 7 1

3 4 8 9

using System;

class Program

{

static void Main()

{

// Step 1: Read the Size of the Square Array

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

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

// Step 2: Read in the Elements of the 2D Array

string[,] array = new string[size, size];

Console.WriteLine("Enter the elements of the array row by row, separated by spaces:");

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

{

string[] row = Console.ReadLine().Split(' ');

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

{

array[i, j] = row[j];

}

}

// Step 3: Swap Rows and Columns

string[,] newArray = new string[size, size];

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

{

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

{

newArray[j, i] = array[i, j];

}

}

// Step 4: Print the New Array

Console.WriteLine("Output:");

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

{

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

{

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

}

Console.WriteLine();

}

}

}