1. Write a function that takes an integer minutes and converts it to seconds.

**Examples**

convert(5) ➞ 300

convert(3) ➞ 180

convert(2) ➞ 120

\*Validations and format:

Positive integer for input @"^\d+$"

Features:

Numbers validation, integer validation, special format validation

Math operation, data type conversion

Overflow exception – input size validation

Error message interaction – format, size

Tooltip for UX hint, message box for enhanced UX

Page jumping functions

Call function from another class

If,else-if,else

1. Write a function that takes the base and height of a triangle and return its area.

triArea(3, 2) ➞ 3

triArea(7, 4) ➞ 14

triArea(10, 10) ➞ 50

* The area of a triangle is: (base \* height) / 2

\*Validations and format:

Integer or decimal @"^\d+(.\d+)?$"

Features:

Numbers validation, decimal validation, integer validation, special format validation

Overflow exception – input size validation

Error message interaction – format, size

Tooltip for UX hint, message box for enhanced UX

Page jumping functions

Call function from another class

Math operation

If,else-if,else

1. Given two numbers, return true if the sum of both numbers is less than 100. Otherwise return false.

lessThan100(22, 15) ➞ true

// 22 + 15 = 37

lessThan100(83, 34) ➞ false

// 83 + 34 = 117

lessThan100(3, 77) ➞ true

\*Validations and format:

Decimal with negative sign @"^\-?\d+\.?\d+$"

Features:

Numbers validation, **decimal validation with negative sign**, special format validation

Overflow exception – input size validation

Error message interaction – format, size

Tooltip for UX hint, message box for enhanced UX

Page jumping functions

Call function from another class

If,else-if,else

1. Create a function that returns true if a number is a palindrome.

**Examples**

IsPalindrome(838) ➞ true

IsPalindrome(4433) ➞ false

IsPalindrome(443344) ➞ true

\*Validations and format:

Positive integer validation @"^[0-9]+$"

Logic foreach-loop through array, if all values are same for each identical position pair,True.

foreach (char c in inputarray)

{ if (inputarray [i] == inputarray [arrayLength - 1 - i]) { i++; }}

if (i == arrayLength) Palindrome : True else Palindrome : false

Features:

Numbers validation, integer validation, special format validation

Array elements comparison

Overflow exception – input size validation

Error message interaction – format, size

Tooltip for UX hint, message box for enhanced UX

Page jumping functions

Call function from another class

Foreach loop

If, else

1. Given a string, create a function to reverse the case. All lower-cased letters should be upper-cased, and vice versa.

ReverseCase("Happy Birthday") ➞ "hAPPY bIRTHDAY"

ReverseCase("MANY THANKS") ➞ "many thanks"

ReverseCase("sPoNtAnEoUs") ➞ "SpOnTaNeOuS"

\*Validations and format:

(@"[\w+\s\*]")&&(!string.IsNullOrWhiteSpace()

or can just use if not following the example !string.IsNullOrWhiteSpace()

To switch letter cases:

char[] arrayname = datasource.ToCharArray();

for-loop by the arrayname.Length, within loop, use if-logic if(char.IsUpper(arrayname[i])) then arrayname[i]=char.ToLower(arrayname[i]);

Then compile characters back to a string by outsource=string.Join (“”,arrayname); for display

Extended: use = string.Join("", arrayname.Reverse()); if also want to reverse sequence

Features:

String validation, special format validation, **string manipulation**, **letters case reverse**, **\* letters sequence reverse**

Char[] array[i]=char.ToLower/ToUpper(array[i]);

Array.Reverse(arrayname); arrayname.Reverse();

String.Join(“replace interval”, list of objects)

Overflow exception – input size validation

Error message interaction – format, size

Tooltip for UX hint, message box for enhanced UX

Page jumping functions

Call function from another class

For loop

If,else-if,else

1. Write a C# Sharp program to check whether a given number is even or odd.   
   Test Data : 15  
   Expected Output :  
   15 is an odd integer

\*Validations and format:

If-logic to validate the input as a valid integer or decimal @"^\d+(\.)??\d+$"

Even or odd only applies to integer, decimal always odd, just in case user does not know that

Features:

Numbers validation, **decimal or integer validation**, special format validation

Overflow exception – input size validation

Error message interaction – format, size

Tooltip for UX hint, message box for enhanced UX

Page jumping functions

Call function from another class

If,else-if,else

1. Write a C# Sharp program to read roll no, name and marks of three subjects and calculate the total, percentage and division.    
   Test Data :  
   Input the Roll Number of the student :784  
   Input the Name of the Student :James  
   Input the marks of Physics, Chemistry and Computer Application : 70 80 90  
   Expected Output :  
   Roll No : 784  
   Name of Student : James  
   Marks in Physics : 70  
   Marks in Chemistry : 80  
   Marks in Computer Application : 90  
   Total Marks = 240  
   Percentage = 80.00  
   Division = First

\*Validations and format:

Roll # : natural numbers only @"[1-9]"

Name: letters with space in between if more than one word, only other character allowed is dot by the end of the string for Jr. Sr., etc. @"^[a-zA-z]+(\s[a-zA-z]+)\*\.?$")

Marks: numbers between 0 and 100, decimal allowed with up to two-digit precision @"^\d+(\.[0-9][0-9]?)?$"

Features:

Numbers validation, decimal validation, natural numbers validation, special format validation, **decimal precision validation**, **string validation**

Math operations

Overflow exception – input size validation

Error message interaction – format, size

Tooltip for UX hint, message box for enhanced UX

Page jumping functions

Call function from another class

StringBuilder

Math.Round(decimal, precision);

If,else-if,else

1. Write a program in C# Sharp to find maximum and minimum element in an array.    
   Test Data :  
   Input the number of elements to be stored in the array :3  
   Input 3 elements in the array :  
   element - 0 : 45  
   element - 1 : 25  
   element - 2 : 21  
   Expected Output :  
   Maximum element is : 45  
   Minimum element is : 21

\*Validations and format:

If-logic implemented for validating format depends on the dot position, dot and number relative position, identify/filter one decimal number from another (using a variable with switching-back-and-forth two values to identify if need to set the current number as a new decimal number).

Decimals format here as when dot and number combination:

at the beginning of the string, after a space, or after a non-digit character, or

following a decimal, this combination will be considered as a new decimal number.

Examples:

.5ds 1.2.3.67!..4.56.d12 will be

0.5, 1.2, 0.3, 0.67, 0.4, 0.56, 12

Features:

Numbers validation, **decimal validation**, integer validation, **special format/pattern validation**

Number filter, **decimal numbers filter from a string**

Overflow exception – input size validation

Error message interaction – format, size

Tooltip for UX hint, message box for enhanced UX

Page jumping functions

Call function from another class

StringBuilder

ArrayList

Decimal array

For loop

If,else-if,else