**C# & .NET**

**Exercises**

1. **Introduction**
2. Read from keyboard two numbers. Display the sum.
3. Read from keyboard n numbers, until the number ends in two zeros. Store the numbers and then display them.
4. **Syntax**
5. Interchange the content of two integer numbers read from the keyboard. Display the numbers before and after the interchange.
6. Resolve the first-grade equation. a\*x+b = 0, a!=0
7. **Core C#**
8. Write a program in C# Sharp to count a total number of alphabets, digits and special characters in a string.

Expected Output :

Number of Alphabets in the string is : 21

Number of Digits in the string is : 1

Number of Special characters in the string is : 4

1. Write a C# Sharp program that takes distance and time as input and displays the speed in kilometres per hour and miles per hour.   
   *Test Data:*  
   Input distance(metres): 50000  
   Input timeSec(hour): 1  
   Input timeSec(minutes): 35  
   Input timeSec(seconds): 56  
   *Expected Output:*  
   Your speed in metres/sec is 8.686588  
   Your speed in km/h is 31.27172  
   Your speed in miles/h is 19.4355
2. Write a C# Sharp program that takes a decimal number as input and displays its equivalent in binary form.   
   *Test Data:*  
   Number to convert: 25  
   *Expected Output:*  
   Binary: 11001
3. Write a program in C# that asks the user for two numbers and one operation (+, -, x, /) then calculate the operation and display the result on the screen.

Show the text *Unrecognized character* if the operation symbol is different from the previous ones.

You should use the switch block.

1. Write a method that checks if given number (positive integer) contains digit 3. Do not convert number to other type. Do not use built-in functions like Contains(), StartsWith(), etc.

Expected input and output

IfNumberContains3(7201432) → true

IfNumberContains3(87501) → false

1. Given two integers, write a method that returns first number raised to the power of second one.

Expected input and output

ToThePowerOf(2, 3) → 8

ToThePowerOf(5, 5) → 3125

1. **Arrays and Tuples**
2. Write a program in C# Sharp to find the sum of all elements of the array.   
   Test Data :  
   Input the number of elements to be stored in the array :3  
   Input 3 elements in the array :  
   element - 0 : 2  
   element - 1 : 5  
   element - 2 : 8  
   *Expected Output* :  
   Sum of all elements stored in the array is : 15
3. Write a program in C# Sharp to separate odd and even integers in separate arrays.   
   Test Data :  
   Input the number of elements to be stored in the array :5  
   Input 5 elements in the array :  
   element - 0 : 25  
   element - 1 : 47  
   element - 2 : 42  
   element - 3 : 56  
   element - 4 : 32  
   *Expected Output*:  
   The Even elements are:  
   42 56 32  
   The Odd elements are :  
   25 47
4. Write a program in C# Sharp for addition of two Matrices of same size.   
   Test Data :  
   Input the size of the square matrix (less than 5): 2  
   Input elements in the first matrix :  
   element - [0],[0] : 1, element - [0],[1] : 2, element - [1],[0] : 3, element - [1],[1] : 4  
   Input elements in the second matrix :  
   element - [0],[0] : 5, element - [0],[1] : 6, element - [1],[0] : 7, element - [1],[1] : 8

*Expected Output*:  
The First matrix is:  
1 2  
3 4  
The Second matrix is :  
5 6  
7 8  
The Addition of two matrix is :  
6 8  
10 12

1. **Strings and Regex**
   1. Given two strings, write a method that returns one string made of two strings. First letter of new string is first letter of first string, second letter of new string is first letter of second string and so on.

**Expected input and output**

MixTwoStrings("aaa", "BBB") → "aBaBaB"

MixTwoStrings("good one", "111") → "g1o1o1d one"

* 1. Given a string, write a method that checks if consists of letters only and ends with period. If string has more than one word, words are separated by space.

**Expected input and output**

AlmostOnlyLetters("She is nice.") → true AlmostOnlyLetters("true 222.") → false

* 1. Given a string, write a method that checks if contains decimal digit and if yes returns its value and position. (use \d for digits)

**Expected input and output**

DecimalDigitInformation("This is 9") → "Digit 9 at position 8"

DecimalDigitInformation("ABCdef") → "No digit found!"

1. **Operators and Casts**

Write a class ‘Rational’ that will implement rational numbers operations. (display, format, +, -, \*, /, <, >, >=,<=,==,!=)

1. **Classes and Objects**

* Define a class with two fields: x and y representing the coordinates of a point in the plane.
* Define constructors for initialization and copy for the class Point.
* Write a method to read and write objects of type Point.
* Write a method to calculate the distance between two points.
* Write a class name Triangle with 3 defined fields for type Point.
* Write constructors for the class.
* Write methods to read and write objects of type Triangle.
* Write methods for calculating the perimeter and area of the triangle.

1. **Inheritance**

* Create a C# program that prompts the user for three names of people and stores them in an array of Person-type objects. There will be two people of the Student type and one person of the Teacher type.
* To do this, create a Person class that has a Name property of type string, a constructor that receives the name as a parameter and overrides the ToString () method.
* Then create two more classes that inherit from the Person class, they will be called Student and Teacher. The Student class has a Study method that writes by console that the student is studying. The Teacher class will have an Explain method that writes to the console that the teacher is explaining. Remember to also create two constructors on the child classes that call the parent constructor of the Person class.
* End the program by reading the people (the teacher and the students) and execute the Explain and Study methods.
* Input
  + Juan
  + Sara
  + Carlos
* Output
  + Explain
  + Study
  + Study

1. **Generics**

* Write a C# program that contains a generic method to Swap two variable.

E.g. Before swap: a=2; b=3

After swap: a=3; b=2

Before swap: a=“abc”; b=“def”

After swap: a=‘def’; b=‘abc’

Collections

* Create a class named Employee with properties (EmployeeId, EmployeeFirstName, EmployeeLastName, EmployeeAge).
* Create a list of Employees and : add employees in it, remove employees, display employees.

1. **LINQ**

* Create
  + a class named Person with properties: PersonId, FirstName, LastName, Age, RoleId
  + a class named Role with properties: RoleId, RoleDescription
* Create
  + 5 objects of type Person and store them in a list
  + 5 objects of type Role and store them in a list
* Do the following operations:
  + Get all the people with age > 18
  + Get all the person`s name
  + Get all people with RoleId=1
  + Get the first two youngest people (take)
  + Get all but most two youngest people (skip)
  + Get all people with their role
  + List of people order by age descending
  + List of people group by role
  + Person with first name ‘FirstName1’
  + Total of people with role ‘Leader’
  + Check if there is any ‘BA’
  + First person with age 60
* Create a class named
  + Product with properties: ProductId, Name, UnitPrice, Category (Food, Drink, Care, Other)
  + Customer with properties CustomerId, CustomerName, Country, City, Phone
  + Order with properties: OrderId, OrderDate, CustomerId
  + OrderDetails with properties: OrderId, ProductId, Quantity
* Do the following operations:
  + Get all products with the Unit Price >=10
  + Get all product names
  + Get a list of product names, unit prices where Unit Price >=10
  + Get the list of most 2 expensive products (Take)
  + Get a list of all but most expensive 2 products (Skip)
  + Get a list of all customers with their orders
  + Get a list of products ordered by Category and descending Unit Price
  + Get a list of products grouped by Category
  + Get a list of Customers that have orders in 2021
  + Get the **first** customer that has the phone number “0723 456 789”)
  + Get the total number of customers from London
  + Check if there is any customer from Romaina
  + Get the total sum of the orders from 2021

1. **Errors and exceptions**

* Create a class named Temperature. Crete a method inside the class that displays the temperature only if it is not zero. If the temperature is 0, the method should throw a user defined error : TempIsZeroException.
* Create an array of integers. Ask the user which array element he wants to see. Output the integer that the user asks for. Provide a way for the user to indicate whether he wants another integer or the end of the program. Provide a handles that deals with invalid input. Handle two specific errors: IndexOutOfRangeException and FormatException.

1. **Unit Testing**

* Create a project named Bank;
* Add properties: CustomerName, Balance
* Add constructors
* Add 2 methods: Debit and Credit
* Create a unit test project named BankTest
* Add reference to Bank project
* Write tests for Debit and Credit methods for different situations: Debit\_WithValidAmount, Debit\_WhenAmountIsLessThanZero, Debit\_WhenAmountIsMoreThanBalance;
* Refactor the code if needed (add ifs, try-catch)

1. **Windows Forms**

* Create a basic calculator.
* Note: for evaluating the expression you can use DataTable class, Compute method: <https://docs.microsoft.com/en-us/dotnet/api/system.data.datatable.compute?view=net-5.0>