1. Tenemos entonces el siguiente fragmento de código:

```
int var1 = 2;
int var2 = 5;
int var3 = -4;
int var4 = 3;
int resultado1 = var1 + var2 * var3 – var4;
int resultado2 = var1 + (var2 * var3) – var4;
int resultado3 = (var1 + var2) * var3 – var4;
int resultado4 = var1 + var2 * (var3 – var4);
if (resultado1 == resultado2)
        Console.WriteLine(“Uno y dos son iguales”);
if (resultado1 == resultado3)
        Console.WriteLine(“Uno y tres son iguales”);
if (resultado1 == resultado4)
        Console.WriteLine(“Uno y cuatro son iguales”);
```

¿Cuánto valen **resultado1**, **resultado2**, **resultado3** y **resultado4**? ¿Qué se imprime en la pantalla? ¿Por qué? Dé una breve explicación.

RTA:

El resultado1 es igual a -21 porque primero se empieza por las operaciones de mayor precedencia que sería la multiplicación, ósea 5 x (-4) = -20. Luego seguiría hacer el resultado anterior sumado a la var1 y quedaría: 2+(-20) = -18. Y finalmente hacemos la resta entre lo que obtuvimos con 3, quedando así: **-21**

El resultado2 es igual a -21, acá se enfatiza o se provoca a que se resuelva lo que está entre paréntesis, cosa que es redundante porque el compilador es a lo primero que va a atacar. El procedimiento es el mismo que el anterior

El resultado3 es igual a -31, primero se realiza lo que esta entre paréntesis dando como resultado 7, luego a eso por orden de precedencia lo multiplicamos con var3, y quedaría 7*(-4), ósea -28. Y finalmente a lo que obtuvimos hay que restarle 3. Y queda -31

El resultado4 es igual a -33, primero se hace lo de paréntesis dando como resultado -7, a eso ultimo lo multiplicamos por la var2 dando así, -35. Y al final a -35 le sumamos 2, quedando -33.

Lo que se imprime por pantalla es el primer if, porque la condición planteada da verdadero y la explicación ya fue dada.

2. Suponga que **x1** y **x2** son dos variables de tipo **bool** (o sea, sólo pueden tomar valores **true** o **false**.) Tenemos el siguiente fragmento de código:

```
if ((!x || y) && x)
    Console.WriteLine("uno");
else
    Console.WriteLine("dos");
```

¿Cuál (o cuáles) de las siguientes afirmaciones es correcta? Justifique brevemente su respuesta.

a) La salida por pantalla "**uno**" sólo ocurre cuando los valores de **x** e **y** son diferentes.
b) La salida por pantalla "**uno**" sólo ocurre cuando los valores de **x** e **y** son iguales.
c) La salida por pantalla "**uno**" sólo ocurre cuando **x** es **true**.
d) La salida por pantalla "**uno**" sólo ocurre cuando **x** es **false**.

Voy a hacer la tabla de la verdad para hacer más visual la explicación

| X | Y | ¡X (lo opuesto) | ¡X\|\|Y (en la disyunción obtenemos falso si ambas proposiciones son falsas) | (¡X\|\|Y)&&X (en la conjunción si ambas prop. Son verdaderas, el resultado es V) | |
|---|---|---|---|---|---|
| V | V | F | V | V | |
| V | F | F | F | F | |
| F | V | V | V | F | |
| F | F | V | V | F | |

Ahora analicemos item por item:

- La a) es falsa porque tanto X como Y deben tener el mismo valor para que se cumpla o de verdadera la condición lógica final
- La b) es verdadera, ya que se cumple y por demostración de la tabla de verdad es correcto.
- La c) es verdadera, pero un poco ambigua porque el resultado final no va a depender 100% de que sólo una de las 2 proposiciones sea verdadera.
- La d) es incorrecta, si x es falso, obtendremos un F

3. Escriba un programa en C# que calcule, para un arreglo de enteros de ***n*** posiciones, la suma de todos los números divisibles por **5** del arreglo y muestre el resultado por pantalla.
A continuación, pruebe su programa con el arreglo **[10, 2, -3, 5, 6, -5, 2, -5, 7, 5, 1, 5]**.

> Recuerde que para determinar si un número es divisible por **5** o no, se puede comparar el resto de la división por **5** con **0**: si **(x % 5 == 0)**, entonces **x** es un número divisible por **3**; en caso contrario, no lo es.

```
using System;
using System.Collections.Generic;
public class Hello{
    static void mostrarVector(int[] vector)
    {
        for(int i = 0; i< vector.Length; i++)
        {
            Console.WriteLine(vector[i] + " ");
        }
    }

    static bool esDivisiblePor5(int numero)
    {
        if(numero % 5 == 0)
        {
            return true;
        }
        return false;
    }

    static int sumandoLosDivisiblesPor5(int[] vectorDeNumeros)
    {
        int sumadorDeDivPor5 = 0;
        for(int i = 0; i<vectorDeNumeros.Length; i++)
        {
            if(esDivisiblePor5(vectorDeNumeros[i]))
            {
                sumadorDeDivPor5 += vectorDeNumeros[i];
            }
        }
        return sumadorDeDivPor5;
    }
    public static void Main(){
        // Your code here!
        int[] testingNumeros = {10, 2, -3, 5, 6, -5, 2, -5, 7, 5, 1, 5};
        int respuesta = 0;

        respuesta = sumandoLosDivisiblesPor5(testingNumeros);
        Console.WriteLine(respuesta);
    }
}
```

4. Escriba un programa que tome como entrada dos arreglos de enteros y determine si el primer arreglo es un subconjunto del segundo.

> Por ejemplo, el arreglo **A = [1, 2, 2, 3, 1]** es un subconjunto del arreglo **B = [1, 2, 3, 4]**, pues todos los números de ***A*** están en ***B***. De la misma manera, el arreglo **A = [1, 1, 1, 1, 1]** es un subconjunto del arreglo **B = [1]**. En cambio, el arreglo **A = [3, 5]** no es un subconjunto del arreglo **B = [1, 2, 3, 4]**.

```
using System;
using System.Collections.Generic;

public class Hello{

    static bool esSubconjunto(int[] A, int[] B)
    {
        int cantidadDeAciertos = 0;
        for(int i=0; i<A.Length; i++) //El subconjunto generalmente es el más chico o igual al
conjunto más grande
        {
            for(int j=0; j<B.Length; j++)
            {
                if(A[i] == B[j])
                {
                    cantidadDeAciertos++;
                }
            }
        }

        //Ahora si la cantidad de aciertos coincide con la cantidad total de elementos del
subconjunto es porque lo es
        if(cantidadDeAciertos == A.Length)
        {
            return true;
        }
        return false;
    }

    public static void Main(){
        // Your code here!
        int[] conjuntoPadre = {10, 2, 3, 4, 100, 5, 1000};
        int[] subconjunto = {10, 100, 1000};

        bool respuesta = false;

        respuesta = esSubconjunto(subconjunto, conjuntoPadre);

        Console.WriteLine(respuesta);
    }
}
```

5. Escriba un programa en C# que reciba una lista de números enteros e imprima todos los números que sean menores que vecino izquierdo y mayores que su vecino derecho. En el caso de los números inicial y final, basta con que sea mayor que su vecino derecho (el inicial) y menor que su vecino izquierdo (el final.)

Algunos ejemplos:

- Si la lista es **[1, 3, 2, 1, 3, 0, 2]**, entonces el programa debe imprimir **1** (el número marcado en rojo.)
- Si la lista es **[7, 5, 3, 1]**, entonces el programa debe imprimir todos los números.
- Si la lista es **[1, 1, 1]**, entonces el programa no debe imprimir nada.

```
using System;
using System.Collections.Generic;

public class Hello{

    static void mostrarLista(List<int> A)
    {
        Console.Write("<");
    for (int i = 0; i < A.Count; i++)
    {
        Console.Write(A[i]);
    if (i < A.Count - 1)
        Console.Write(", ");
    else
        Console.WriteLine(">");
    }
    }


    static List<int> ejercicio5(List<int> lista)
    {
        List<int> resultado = new List<int>();
        for(int i = 0; i< lista.Count; i++)
        {
            if(i==0)
            {
                if(lista[i]>lista[i+1])
                {
                    resultado.Add(lista[i]);
                }
            }
            if(i== (lista.Count-1))
            {
                if(lista[i]<lista[i-1])
                {
                    resultado.Add(lista[i]);
                }
            }

            if(i>0 && i<lista.Count-1)
            {

             if((lista[i]<lista[i-1]) && (lista[i]>lista[i+1]))
            {
                resultado.Add(lista[i]);
            }
            }

        }

        return resultado;
```

```
    }

    public static void Main(){
        // Your code here!
        List<int> A = new List<int> {7,5,3,1};
        mostrarLista(ejercicio5(A));
        //mostrarLista(A);
    }
}
```

6. Escriba un método externo que elimine de una lista de números enteros todos los números que sean mayores a **7** y menores a **15**. Escriba dos versiones de su método:
   a) Utilizando ***RemoveAll***.
   b) Sin utilizar ***RemoveAll***.

Acá van los métodos:

```
static bool esMayorA7YMenorA15(int num)
    {
        if(num > 7 && num < 15)
        {
            return true;
        }
        return false;
    }

    static void eliminadoLosMayoresA7YMenoresA15ConRemoveAll(List<int> A)
    {
        A.RemoveAll(esMayorA7YMenorA15);
    }

    static void eliminadoLosMayoresA7YMenoresA15SinRemoveAll(List<int> A)
    {
        for(int i = 0; i< A.Count; i++)
        {
            if(esMayorA7YMenorA15(A[i]))
            {
                A.Remove(A[i])
            }
        }
    }
```