Programming Fundamentals

Exercises

Dr. Enric Sesa i Nogueras

TecnoCampus
Escola Superior Politècnica

INTRODUCTION
VERY FIRST STEPS
SIMPLE PROGRAMS

VERY FIRST STEPS. SIMPLE PROGRAMS

0a **[LOOKING TOWARDS THE FUTURE]** In order to acquire some familiarity with the programming environment, write and execute the following program:

```
static void Main(string[] args)
{
    string name;
    int age, currentYear;

    Console.WriteLine("LOOKING TOWARDS THE FUTURE");
    Console.WriteLine("--------------------------");
    Console.WriteLine();

    Console.Write("What's your name? ");
    name = Console.ReadLine();

    Console.Write("{name}, what year is this? ", name);
    currentYear = Convert.ToInt32(Console.ReadLine());

    Console.Write("{name}, how old are you? ", name);
    age = Convert.ToInt32(Console.ReadLine());

    Console.WriteLine();
    Console.WriteLine("{name}, if everyting goes ok, in year {currentYear + 10}, you'll be {age + 10}");

    Console.WriteLine();

    Console.Write("Press any key to exit");
    Console.ReadKey();
}
```

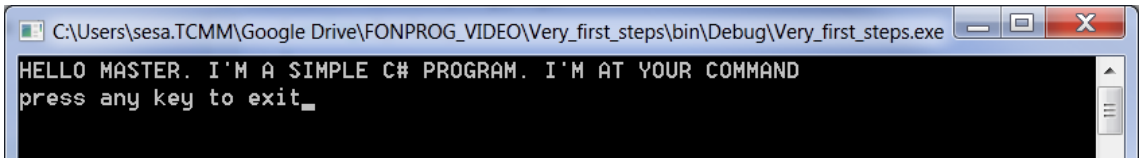
0b. **[SPEHRE CALCULATIONS]** The following program was conceived to determine the surface and the volume of a sphere given its radius. It is logically correct but contains several syntax errors. Fix the syntax errors so that the program can be run. $S = 4 \cdot \pi \cdot r^2$; $V = 4/3 \cdot \pi \cdot r^3$

The image below shows the expected outcome for r=2

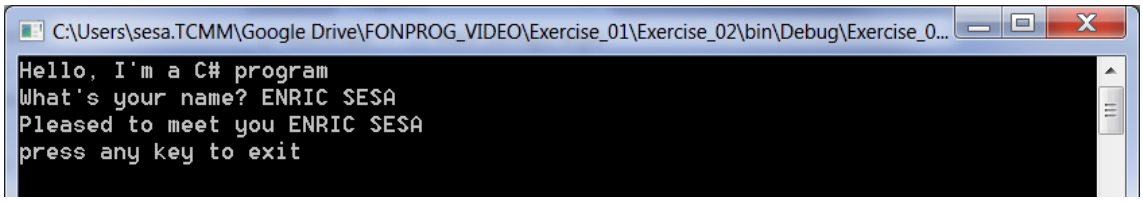
```
public static void Main(string[] args)
{

    String userName;
    double radius;
    double surface, volume, litres;
    double pi;

    /* REMINDERS:
     * The surface of a sphere of radius r is 4·pi·r·r
     * The volume of a sphere of radius r is (4/3)·pi·r·r·r
     * There are 1000 litres in a cubic meter
    */

    pi = 3.141516927;

    Console.WriteLine("    SPHERE CALCULATIONS");
    Console.WriteLine("    ------ ------------");

    Console.WriteLine();

    Console.write("Please, write your name ");
    userName = Console.ReadLine();
    Console.Writeline();

    Console.Write($"Please {userName}, write the radius of the sphere (in metres) ");
    radius = convert.ToDouble(Console.ReadLine());

    surface = 4.0 * pi * radius * radius

    volume = 4.0 \ 3.0 * pi * radius * radius * radius;
    litres = volume * 1000.0;

    Console.WriteLine();
    Console.WriteLine($"Look {userName, a sphere with radius {radius} metres");
    Console.WriteLine($"   Has surface of surface} square metres");
    Console.WriteLine("   and a volume of {volume} cubic metres, equivalent to {litres} litres");

    Console.writeLine();
    Console.write("Press any key to continue");
    console.Readkey();
}
```

```
    SPHERE CALCULATIONS
    ------ ------------

Please, write your name ENRIC

Please ENRIC, write the radius of the sphere (in metres) 2

Look ENRIC, a sphere with radius 2 metres
   Has surface of 50.264270832 square metres
   and a volume of 33.509513888 cubic metres, equivalent to 33509.513888 litres

Press any key to continue
```

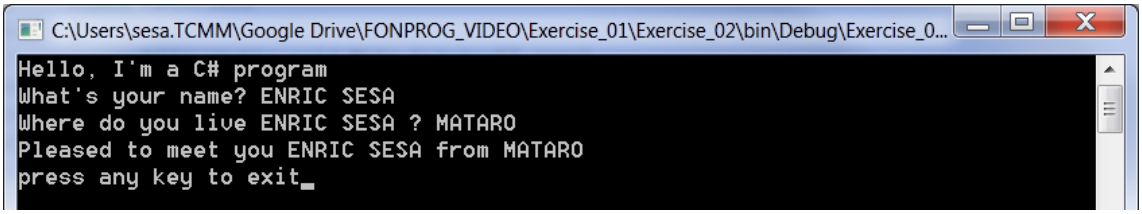
1. Write, compile and execute a program that shows the sentence "**HELLO MASTER. I'M A SIMPLE C# PROGRAM. I'M AT YOUR COMMAND**" and then waits until a key is pressed.



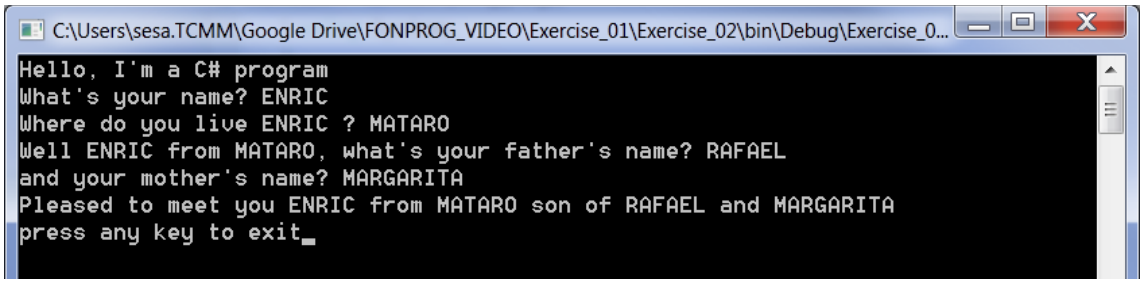
2. Write, compile and execute a program that...
   a. "Asks" your name, greets you and then waits until a key is pressed. You'll need a String variable to store the name.



   b. Extend the program so that it also asks the name of the place where you live. You'll need another String variable to store the name of the place where you live



   c. Extend the program again. Now it has to ask the names of your parents (father and mother). You'll need two more variables...



You can try to solve Exercise 2 with and without placeholders.

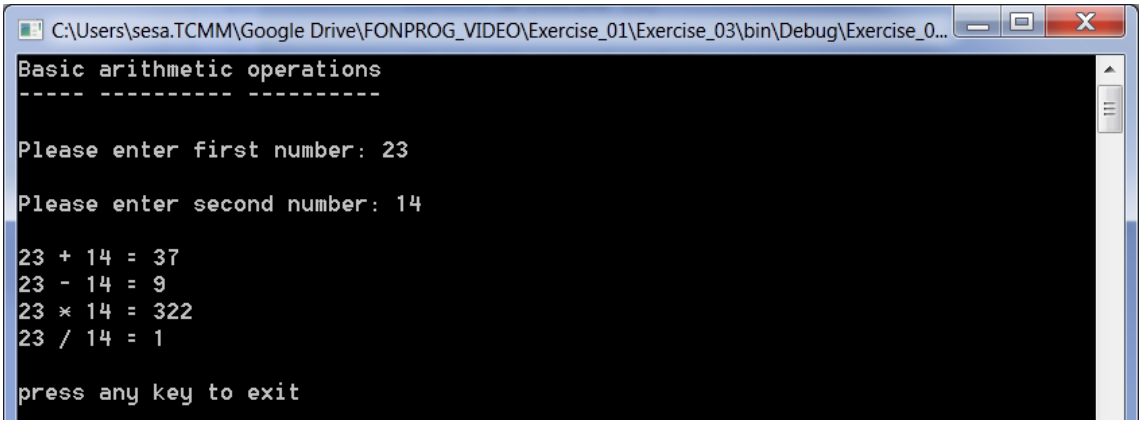
See solution of previous exercise (2c) on next page.

```
// SOLUTION WITH PLACEHOLDERS (INJECTION)
public static void Main (string[] args)
{
    String yourName;
    String place;
    String father, mother;

    Console.WriteLine ("Hello, I'm a C# program");
    Console.Write ("What's your name? ");
    yourName = Console.ReadLine ();

    Console.Write($"Where do you live {yourName} ? ");
    place = Console.ReadLine ();

    Console.Write($"Well {yourName} from {place}, what's your father's name? ");
    father = Console.ReadLine ();
    Console.Write ("and your mother's name? ");
    mother = Console.ReadLine ();

    Console.WriteLine($"Pleased to meet you {yourName} from {place} son of {father} and {mother}");

    Console.Write ("press any key to exit");
    Console.ReadLine ();
}
```

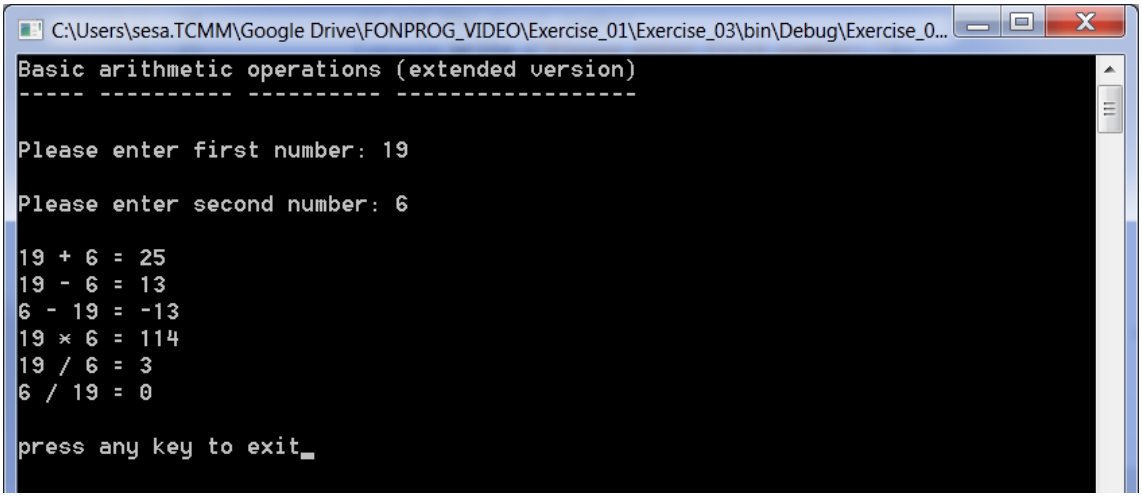
3.

a. The purpose of the following incomplete program is to compute the addition, the subtraction, the multiplication and the division of two integer numbers. Complete it by inserting the necessary code after the **/* COMPLETE */** comments

```
public static void Main (string[] args)
{
    int number1, number2;
    // these two variables will store the numbers that have to be operated

    int addition, subtraction, multiplication, division;
    // this four variables will store the results of the operations

    Console.WriteLine ("Basic arithmetic operations");
    Console.WriteLine ("----- ---------- ----------");
    Console.WriteLine ();

    // get the first number
    Console.Write ("Please enter first number: ");
    number1 = Convert.ToInt32 (Console.ReadLine());
    Console.WriteLine ();

    // get the second number
    /* COMPLETE */

    addition = number1 + number2;  // compute addition and store in variable
    /* COMPLETE */

    // write the results
    Console.WriteLine ($"{number1} + {number2} = {addition}");
    /* COMPLETE */

    Console.WriteLine ();
    Console.Write ("press any key to exit");
    Console.ReadLine ();

}
```



b. Now improve the program so that it also shows number2 – number1 and number2 / number1 (subtraction and division are non-commutative operations).



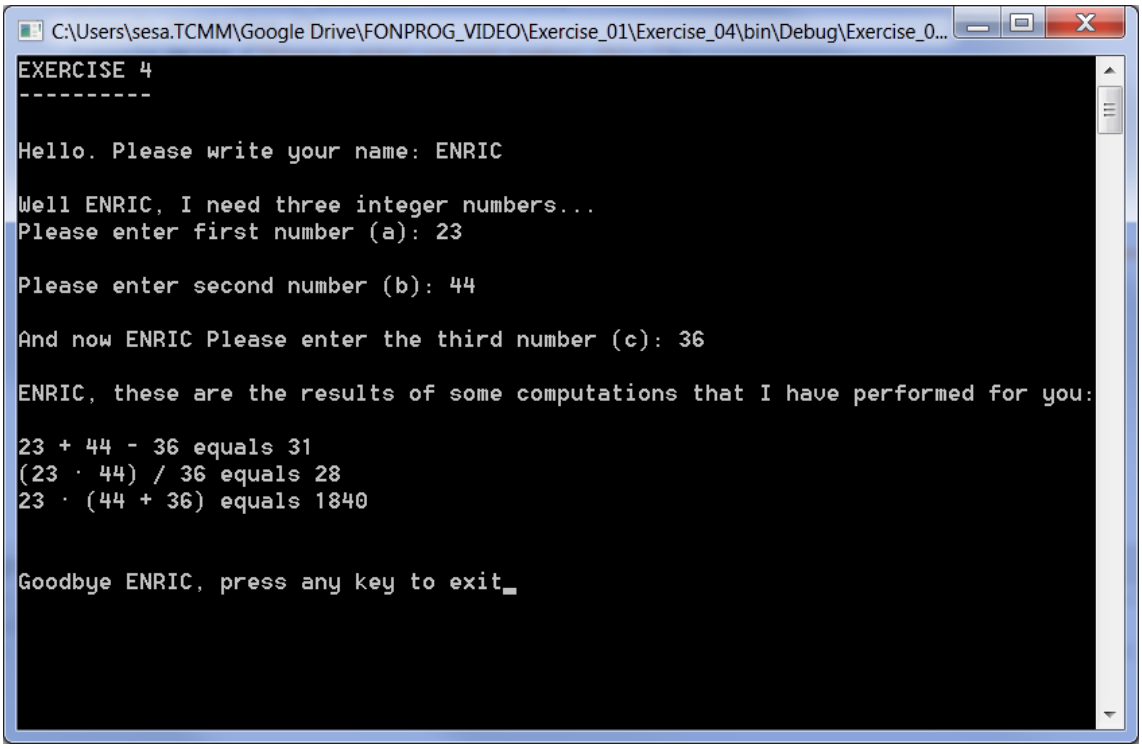
c. Rewrite the program so that instead of using the four variables `addition`, `subtraction`, `multiplication` and `division` it uses just a single variable named `result`.

See solution of previous exercise (3a) on next page.

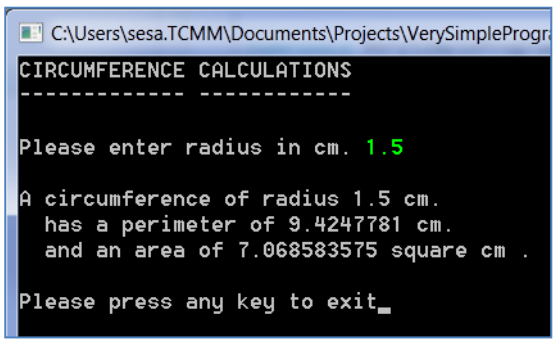
```
// SOLUTION 3a
public static void Main (string[] args)
{
    int number1, number2;
    // these two variables will store the numbers that have to be operated

    int addition, subtraction, multiplication, division;

    Console.WriteLine ("Basic arithmetic operations");
    Console.WriteLine ("----- ---------- ----------");
    Console.WriteLine ();


    // get the first number
    Console.Write ("Please enter first number: ");
    number1 = Convert.ToInt32 (Console.ReadLine());
    Console.WriteLine ();

    // get the second number
    Console.Write ("Please enter second number: ");
    number2 = Convert.ToInt32 (Console.ReadLine());
    Console.WriteLine ();

    addition = number1 + number2;  // compute addition and store in variable
    subtraction = number1 - number2; // same for subtraction
    multiplication = number1 * number2; // same for...
    division = number1 / number2; // same for...


    // write the results
    Console.WriteLine ($"{number1} + {number2} = {addition}");
    Console.WriteLine ($"{number1} - {number2} = {subtraction}");
    Console.WriteLine ($"{number1} * {number2} = {multiplication}");
    Console.WriteLine ($"{number1} / {number2} = {division}");

    Console.WriteLine ();
    Console.Write ("press any key to exit");
    Console.ReadLine ();

}
```

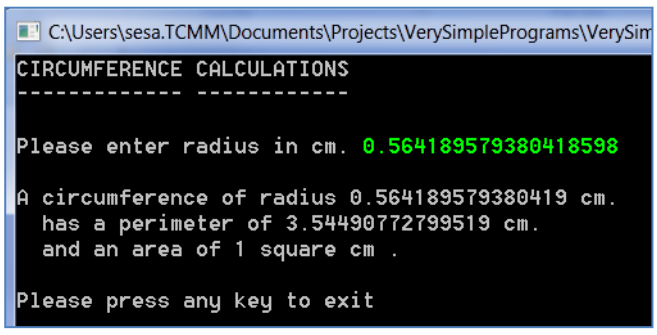
4. Write compile and execute a program that gets the name of its user and three integer numbers (call them a, b and c) and then computes the following operations
   - a + b – c
   - (a · b) / c
   - a · (b + c)

   The output of your program has to be similar to the image shown below. Notice that the name of the user appears several times. Remember that the multiplication operator in C# is not · but *

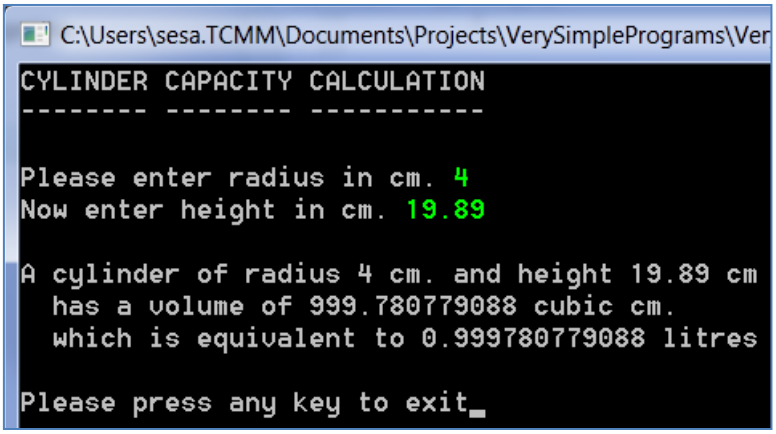


5. Write a program that given the radius (in centimetres) of a circumference computes its perimeter and its area (assume π=3.1415927).
   Perimeter is $2\pi r$; area is πr^2 and r^2 is r·r
   The following images display the results of two different executions of the program.





6. Write a program that given the radius of its base and the height (in centimetres) of a cylinder computes its capacity in litres (assume π=3.1415927. 1 litre = 1000 cm^3). **Capacity is the result of multiplying the area of the base by the height**. The following image displays the result of an execution of the program.



See solutions for exercises 4, 5 and 6 on next pages

```
// Exercise 4. SOLUTION
public static void Main (string[] args)
{
    int a, b, c;
    String name;

    Console.WriteLine ("EXERCISE 4");
    Console.WriteLine ("----------");
    Console.WriteLine ();

    Console.Write ("Hello. Please write your name: ");
    name = Console.ReadLine ();
    Console.WriteLine ();

    Console.WriteLine ($"Well {name}, I need three integer numbers...");
    Console.Write ("Please enter first number (a): ");
    a = Convert.ToInt32 (Console.ReadLine());
    Console.WriteLine ();
    Console.Write ("Please enter second number (b): ");
    b = Convert.ToInt32 (Console.ReadLine());
    Console.WriteLine ();
    Console.Write ($"And now {name} Please enter the third number (c): ");
    c = Convert.ToInt32 (Console.ReadLine());
    Console.WriteLine ();

    Console.WriteLine ($"{name}, these are the results of some computations that I have performed for you:");
    Console.WriteLine ($"{a} + {b} - {c} equals {a+b-c}");
    Console.WriteLine ($"({a} · {b}) / {c} equals {(a*b)/c}");
    Console.WriteLine ($"{a} · ({b} + {c}) equals {a*(b+c)}");
    Console.WriteLine ();

    Console.WriteLine ();
    Console.Write ("Goodbye {0}, press any key to exit",name);
    Console.ReadLine ();
}
```

```
// Exercise 5. SOLUTION
public static void Main (string[] args)
{
    double radius;
    double area, perimeter;

    Console.WriteLine ("CIRCUMFERENCE CALCULATIONS");
    Console.WriteLine ("------------ -----------");
    Console.WriteLine ();

    Console.Write ("Please enter radius in cm. ");
    Console.ForegroundColor = ConsoleColor.Green;
    radius = Convert.ToDouble(Console.ReadLine ());
    Console.ResetColor ();

    area = 3.1415927 * radius * radius;
    perimeter = 2 * 3.1415927 * radius;

    Console.WriteLine ();
    Console.WriteLine ($"A circumference of radius {radius} cm.");
    Console.WriteLine ($"  has a perimeter of {perimeter} cm.");
    Console.WriteLine ($"  and an area of {area} square cm.");
    Console.WriteLine ();

    Console.Write ("Please press any key to exit");
    Console.ReadKey (true);

}
```

```
// Exercise 6. Solution
public static void Main (string[] args)
{
    double radius, height;
    double volume, litres;

    Console.WriteLine ("CYLINDER CAPACITY CALCULATION");
    Console.WriteLine ("-------- -------- -----------");
    Console.WriteLine ();

    Console.Write ("Please enter radius in cm. ");
    Console.ForegroundColor = ConsoleColor.Green;
    radius = Convert.ToDouble(Console.ReadLine ());
    Console.ResetColor ();
    Console.Write ("Now enter height in cm. ");
    Console.ForegroundColor = ConsoleColor.Green;
    height = Convert.ToDouble(Console.ReadLine ());
    Console.ResetColor ();

    volume = 3.1415937 * radius * radius * height;
    litres = volume / 1000;

    Console.WriteLine ();
    Console.WriteLine ($"A cylinder of radius {radius} cm. and height {height} cm");
    Console.WriteLine ($"  has a volume of {volume} cubic cm.");
    Console.WriteLine ($"  which is equivalent to {litres} litres.");
    Console.WriteLine ();

    Console.Write ("Please press any key to exit");
    Console.ReadKey (true);
}
```

7. Write a program that given an integer number of seconds expresses it as days plus hours plus minutes plus seconds.

Remember that there are 60 seconds in a minute, 3600 seconds in an hour and 86400 seconds in a day.

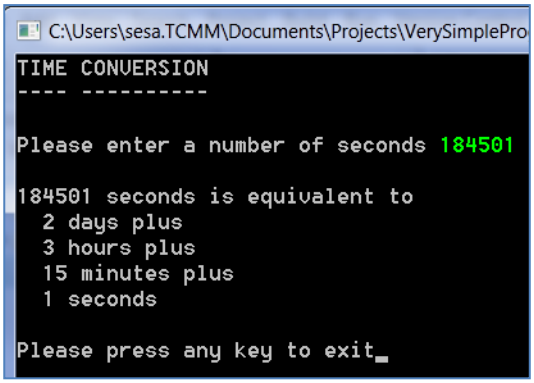
The conversion can be performed this way:

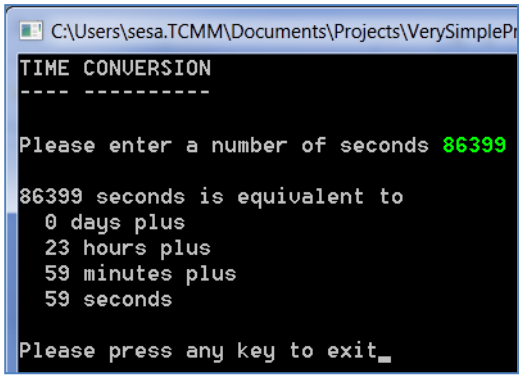
- Divide (integer division) the number of seconds into 86400. The quotient is the number of days; the remainder is a number of seconds less than a day. Take this number of seconds (the remainder) and…
- Divide it into 3600. The quotient is the number of hours; the remainder is a number of seconds less than an hour. Take this number and…
- Divide it into 60. The quotient is the number of minutes; the remainder is the number of seconds less than a minute

It can also be performed this way:

- Divide the number of seconds into 60. The quotient is the number of minutes and the remainder is the final number of seconds
- Divide the number of minutes into 60. The quotient is the number of hours and the remainder is the final number of minutes
- Divide the number of hours into 24. The quotient is the number of days and the remainder is the final number of hours

The following images display the results of two different executions of the program.

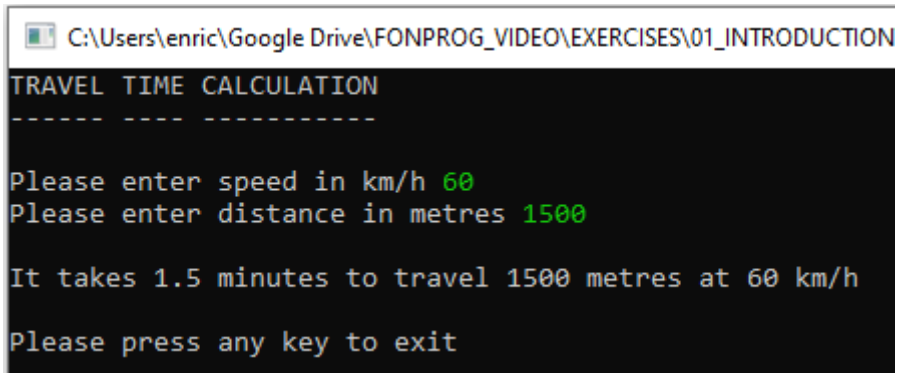


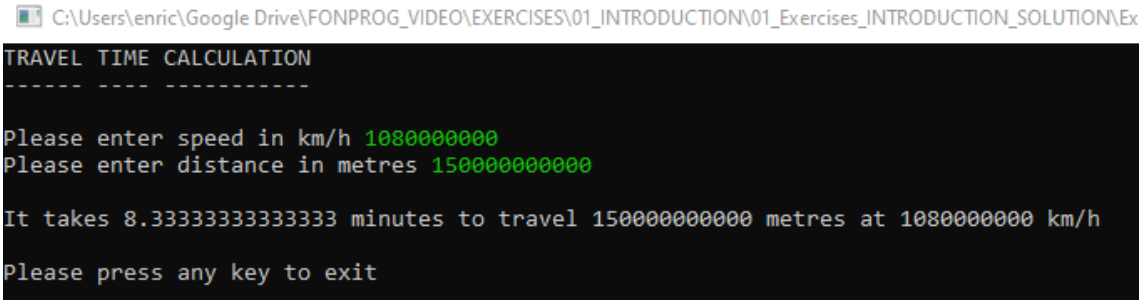


8. Write a program that given a speed in km/h and a distance in metres computes the time required to travel that distance at the given speed (time is distance divided into speed). Express the result in minutes.

Your program should proceed this way:
- Convert speed in kilometres per hour into metres per second. As there are 1000 metres in a kilometre and 3600 seconds in an hour, multiply the speed given by 1000 and divide the result into 3600. You do not need a new variable - Time is distance divided into speed. As distance is in metres and speed in metres per second, the result will be expressed in seconds - Divide time into 60 in order to express it in minutes

The following images display the results of two different executions of the program (the last one is the result of computing the time that light takes to travel from the Sun to the Earth)

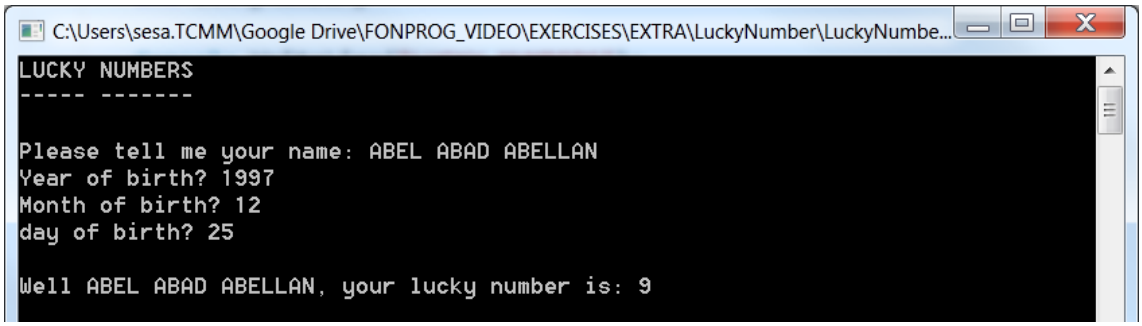




9. **[THE LUCKY NUMBER]** Complete the program that computes your "lucky number". Your lucky number is:
   - the year you were born divided into 100 minus
   - the number of the month you were born plus
   - the day you were born divided into 10

   For instance, if you had been born the 25th of December 1997 your lucky number would be 1997/100 – 12 + 25/10 = 9

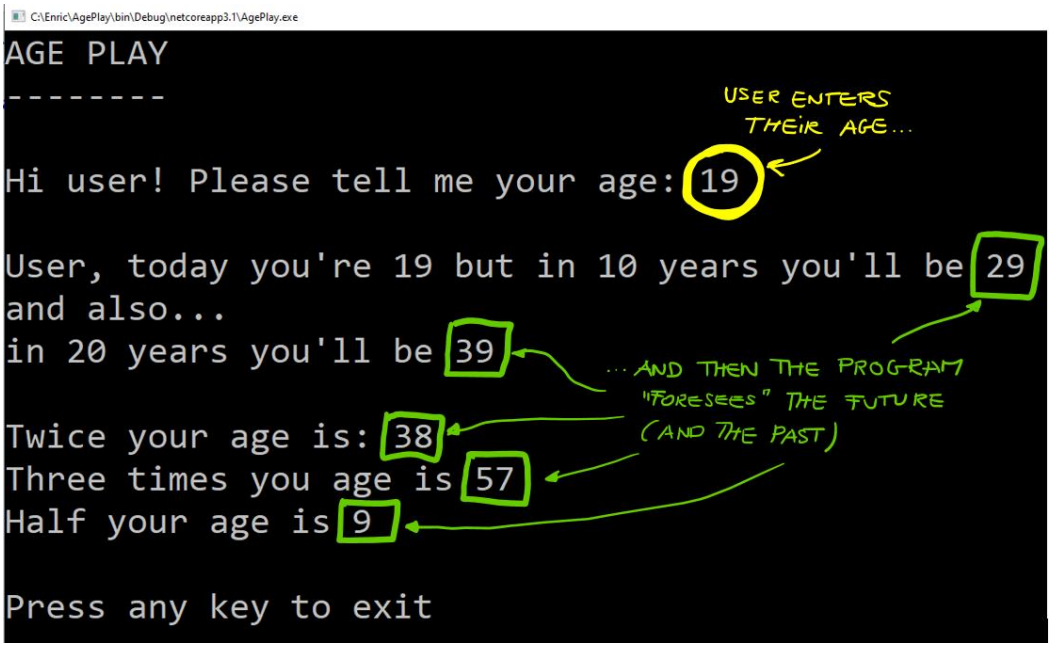
```
public static void Main(string[] args) {
        // DECLARE your variables here
        // one to hold the name,
        // three to hold the year, month and day of birth
        // and one to hold the lucky number
        /* COMPLETE */


        Console.WriteLine("LUCKY NUMBER");
        Console.WriteLine("----- ------");
        Console.WriteLine();

        // get the name, the year, the month and the day...
        Console.Write("Please tell me your name: ");
        name = Console.ReadLine();
        /* COMPLETE */


        // compute the lucky number as follows:
        // divide year into 100; substract month; add day divided into 10
        // save the result in a variable...
        /* COMPLETE */


        // Show the lucky number
        Console.WriteLine();
        /* COMPLETE */


        Console.SetCursorPosition(0, Console.WindowHeight - 1);
        Console.Write("Press any key to exit");
        Console.ReadKey();
}
```

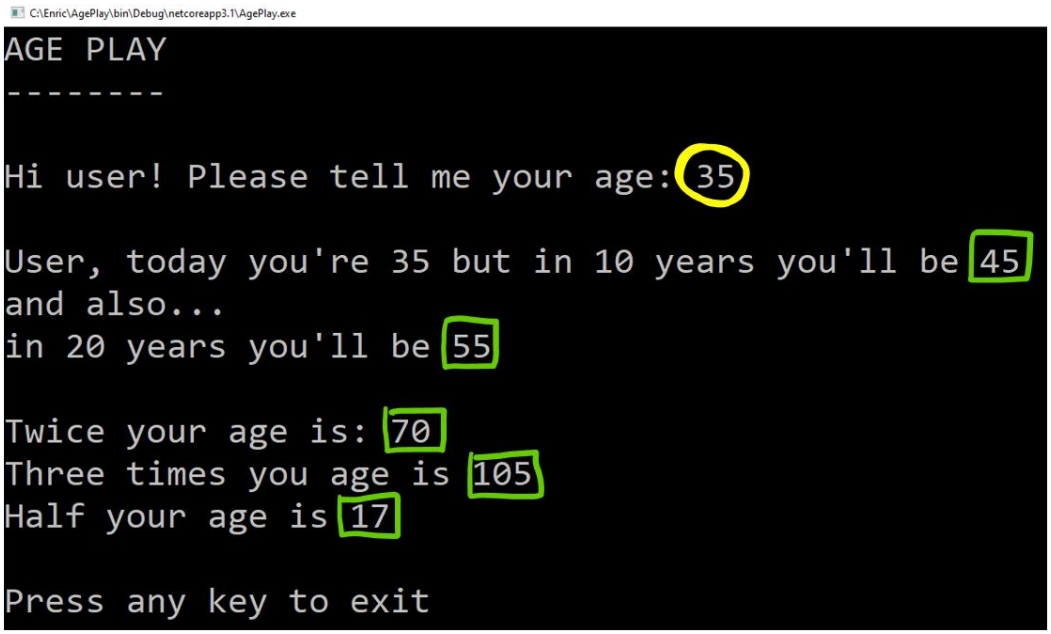


10. **[Age Playing]** Make all the necessary steps to write a program that has the behaviour shown in the following annotated screen captures.

Use the VS Multi Project Solution provided for exercises. As there is no specific project given for this exercise, start by creating one… File → add → new Project … (or in the solution explore add → new Project) Use the Console App template

Three different solutions on next pages Compare them with YOUR solutions





```
static void Main(string[] args)
{
    int age;
    int inTenYears, inTwentyYears, doubleAge, tripleAge, halfAge;

    Console.WriteLine("AGE PLAY");
    Console.WriteLine("--------");
    Console.WriteLine();

    Console.Write("Hi user! Please tell me your age: ");
    age = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine();

    inTenYears = age + 10;
    inTwentyYears = age + 20;
    doubleAge = age * 2;
    tripleAge = age * 3;
    halfAge = age / 2; // beware integer division

    Console.WriteLine($"User, today you're {age} but in 10 years you'll be {inTenYears}");
    Console.WriteLine("and also...");
    Console.WriteLine($"in 20 years you'll be {inTwentyYears}");
    Console.WriteLine();
    Console.WriteLine($"Twice your age is {doubleAge}");
    Console.WriteLine($"Three times your age is {tripleAge}");
    Console.WriteLine($"Half your age is {halfAge}");

    Console.WriteLine();
    Console.Write("Press any key to exit ");
    Console.ReadKey();
}
```

EACH CALCULATION STORED IN A VARIABLE

```
static void Main(string[] args)
{
    int age;
    int newAge;

    Console.WriteLine("AGE PLAY");
    Console.WriteLine("--------");
    Console.WriteLine();

    Console.Write("Hi user! Please tell me your age: ");
    age = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine();

    newAge = age + 10;
    Console.WriteLine($"User, today you're {age} but in 10 years you'll be {newAge}");
    Console.WriteLine("and also...");
    newAge = age + 20;
    Console.WriteLine($"in 20 years you'll be {newAge}");
    Console.WriteLine();
    newAge = age * 2;
    Console.WriteLine($"Twice your age is {newAge}");
    newAge = age * 3;
    Console.WriteLine($"Three times your age is {newAge}");
    newAge = age / 2;
    Console.WriteLine($"Half your age is {newAge}");

    Console.WriteLine();
    Console.Write("Press any key to exit ");
    Console.ReadKey();
}
```

ALL CALCULATIONS STORED IN THE SAME VARIABLE

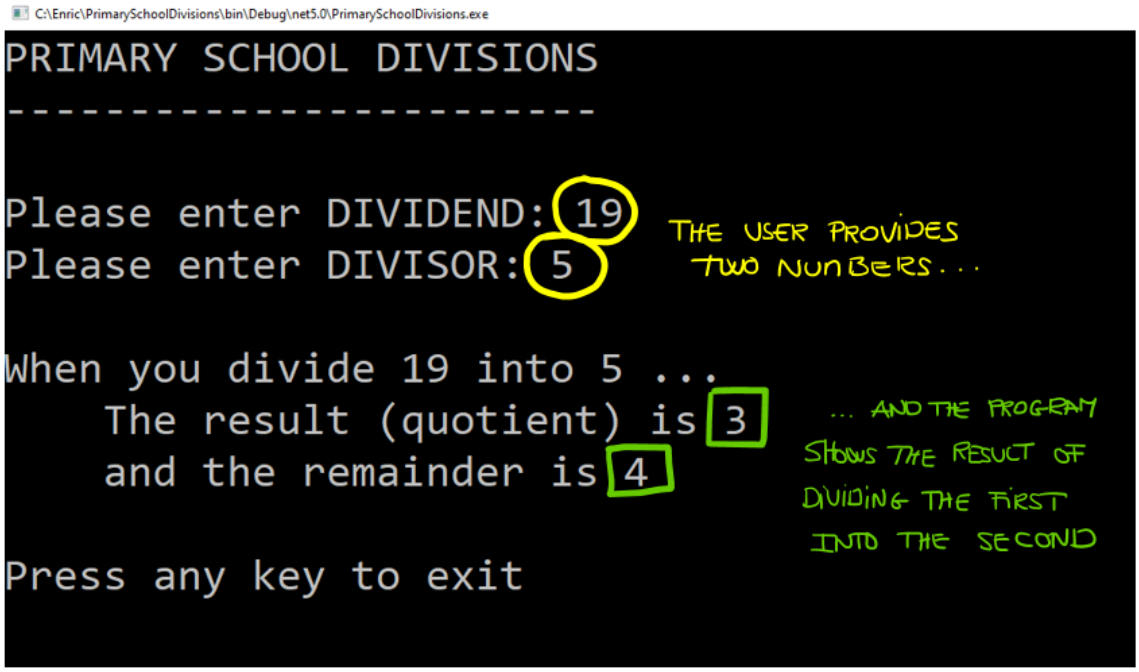
```
static void Main(string[] args)
{
    int age;

    Console.WriteLine("AGE PLAY");
    Console.WriteLine("--------");
    Console.WriteLine();

    Console.Write("Hi user! Please tell me your age: ");
    age = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine();


    Console.WriteLine($"User, today you're {age} but in 10 years you'll be {age+10}");
    Console.WriteLine("and also...");
    Console.WriteLine($"in 20 years you'll be {age+20}");
    Console.WriteLine();

    Console.WriteLine($"Twice your age is {age*2}");
    Console.WriteLine($"Three times your age is {age*3}");

    Console.WriteLine($"Half your age is {age/2}");

    Console.WriteLine();
    Console.Write("Press any key to exit ");
    Console.ReadKey();
}
```

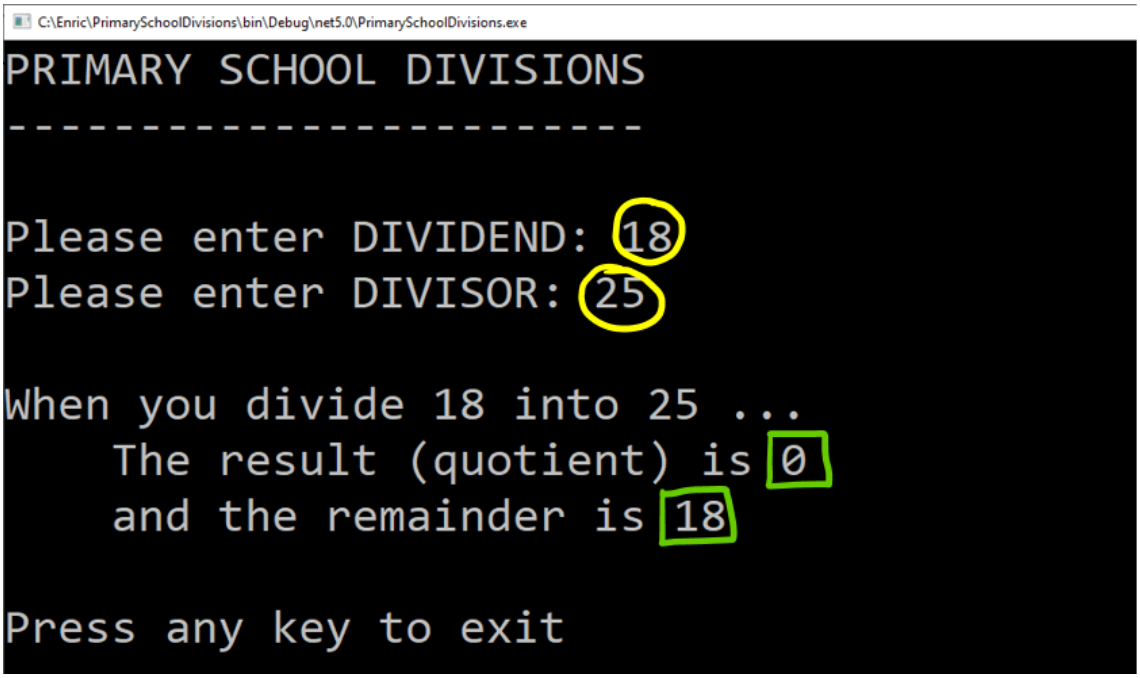
ALL CALCULATIONS WRITTEN WITHOUT BEING STORED

11. **[PRIMARY SCHOOL DIVISIONS]** Make all the necessary steps to write a program that performs "primary school divisions". The expected behaviour is shown in the following annotated screen captures

Use the VS Multi Project Solution provided for exercises. If there is no specific project given for this exercise create one. See comment in previous exercise

Two different solutions on next page





```
// Primary School Divisions. Solution
static void Main(string[] args)
{
    int dividend, divisor;
    int result, remainder;

    Console.WriteLine("PRIMARY SCHOOL DIVISIONS");
    Console.WriteLine("------------------------");
    Console.WriteLine();

    Console.Write("Please enter DIVIDEND: ");
    dividend = Convert.ToInt32(Console.ReadLine());
    Console.Write("Please enter DIVISOR: ");
    divisor = Convert.ToInt32(Console.ReadLine());

    result = dividend / divisor;
    remainder = dividend % divisor;

    Console.WriteLine();
    Console.WriteLine($"When you divide {dividend} into {divisor} ...");
    Console.WriteLine($"    The result (quotient) is {result}");
    Console.WriteLine($"    and the remainder is {remainder}");

    Console.WriteLine();
    Console.Write("Press any key to exit ");
    Console.ReadKey();

}
```

```
// Primary School Divisions. Alternative solution
static void Main(string[] args)
{
    int dividend, divisor;

    Console.WriteLine("PRIMARY SCHOOL DIVISIONS");
    Console.WriteLine("------------------------");
    Console.WriteLine();

    Console.Write("Please enter DIVIDEND: ");
    dividend = Convert.ToInt32(Console.ReadLine());
    Console.Write("Please enter DIVISOR: ");
    divisor = Convert.ToInt32(Console.ReadLine());

    Console.WriteLine();
    Console.WriteLine($"When you divide {dividend} into {divisor} ...");
    Console.WriteLine($"    The result (quotient) is {dividend/divisor}");
    Console.WriteLine($"    and the remainder is {dividend%divisor}");

    Console.WriteLine();
    Console.Write("Press any key to exit ");
    Console.ReadKey();

}
```

EXPRESSIONS. EXPRESSION EVALUATION

Which is going to be the result of each of the following pieces of code? (What is going to be written when the code is executed?)

Solutions are given at the end of the section

Eval 1)

```
int a;
int b;

a = 10;
b = 15;

a = b + 3 + 5 * b;
b = a - b * 2;
a = a + b;

Console.WriteLine("a = {0}, b = {1}", a, b);
```

Eval 2)

```
int a;
int b;

a = 10;
b = 11;

a = b / 4 % 3;
b = 2 + a * 3 / 2 + 1;

Console.WriteLine("a = {0}, b = {1}", a, b);
```

Eval 3)

```
int a;
int b;
int c;

a = 5;
b = 6;

a = 2 + b - a * 2 / 4;
b = a + 2 * b + 1;
c = a % b + b % a + a / b + b / a;

Console.WriteLine("a = {0}, b = {1}, c = {2}", a, b, c);
```

Eval 4)

```
double a;
double b;

a = 30.0;
b = 10.5;

a = a + 10.0 / 3 + (b - 1 / 2.0);
b = -b;

Console.WriteLine("a = {0}, b = {1}", a, b);
```

```
double a;
double b;

a = 30.0;
b = 10.5;

a = a + 10 / 3 + (b - 1 / 2);
b = - b;

Console.WriteLine("a = {0}, b = {1}", a, b);
```

Eval 5)

```
int e;
int f;

e = -(12 + 4 / 3 * 5);
f = -(12 + 4 / (3 * 5));
e = e + 10 * 4 - 3 / 2;
f = (e + 10) * (4 - 3) / 2;

Console.WriteLine("e = {0}, f = {1}", e, f);
```

Eval 6)

```
int e, f;

e = -1 + 2 * 3 / 4 % 5;
f = -(1 + (2 * (3 / (4 % 5))));

Console.WriteLine("e = {0}, f = {1}", e, f);
```

Eval 7)

```
bool m, n;

m = true && false || true && !false;
n = m && true && (false || true) && !(true || false);

m = !n || !m && !(m && !n);
n = true || m && !(false && n);

Console.WriteLine("m = {0}, n = {1}", m, n);
```

Eval 8)

```
bool m, n;
int a, b;

a = 12 + 7 / 3;
b = 13 - 7 % 3;

m = a - b < b - a;
n = a + 2 > b - 2;

m = a + b != 25 + 4 % 3;
n = (a < 4 || b < 19)

Console.WriteLine("a = {0}, b = {1}, m = {2}, n = {3}",a, b, m, n);
```

Eval 9)

```
char car1, car2, car3;
bool bu;

car1 = 'b';
car2 = '3';
car3 = 'F';

bu = (car1 > car2) && (car3 != car1);
bu = bu && !(car3 <= car2);

Console.WriteLine("bu = {0}", bu);
```

Eval 10)

```
char c = 'a';
int a = 12;
bool res;

res = c != 'c' && 'b' >= c;
res = res == !true || a % 5 >= a % 4;

Console.WriteLine("res = {0}", res);
```

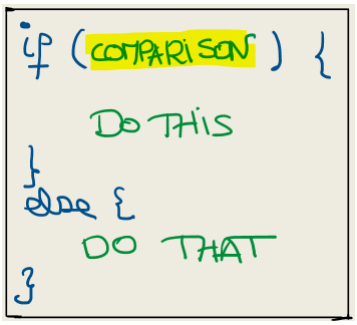
SOLUTIONS

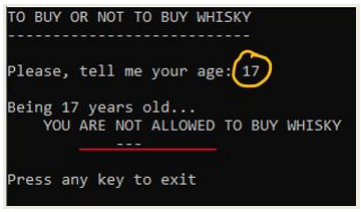
Eval 1) a = 156, b = 63
Eval 2) a = 2, b = 6
Eval 3) a = 6, b = 19, c = 10
Eval 4) first=> a = 43,3333333333333, b = -10,5
second=> a = 43,5, b = -10,5
Eval 5) e = 22, f = 16
Eval 6) e = 0, f = -1
Eval 7) m = True, n = True
Eval 8) a = 14, b = 12, m = False, n = True
Eval 9) bu = True
Eval 10) res = True

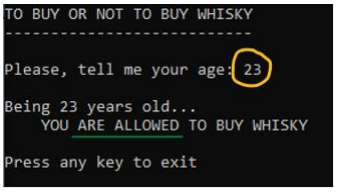
CONDITIONAL EXECUTION

SHORT PROGRAMS

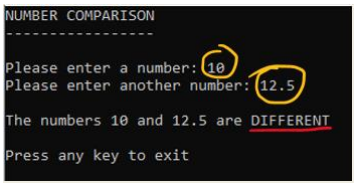
0a. [Can You (legally) buy whisky?] Write a program that asks its user his/her age and then tells him if he/she can buy whisky or not.

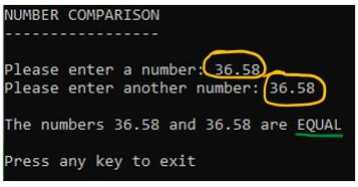




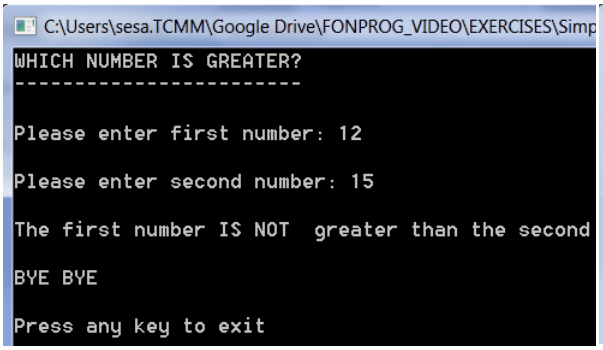


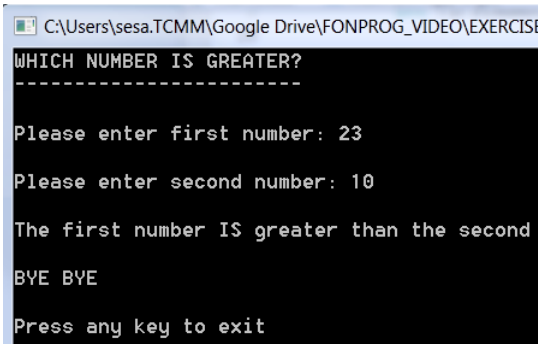
0b. Write a program that gets two double numbers from its user and tells if they are the same or not

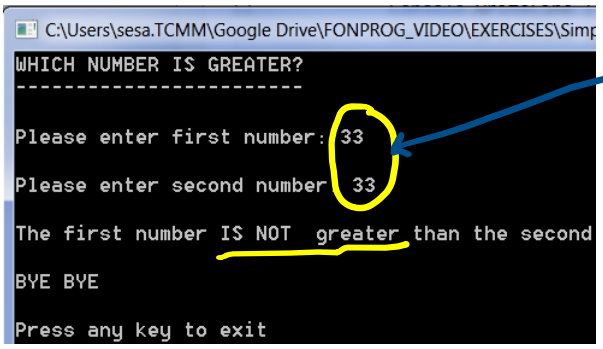




1. Write a program that gets two integer numbers and tells if the first number is greater than the second or not. The program always says "BYE BYE" at the end. Notice that if the two numbers are the same then the program must say that the first is not greater than the second.

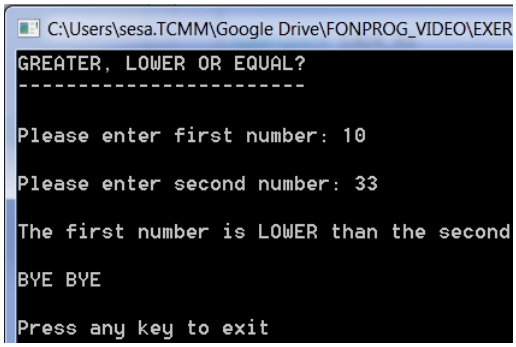


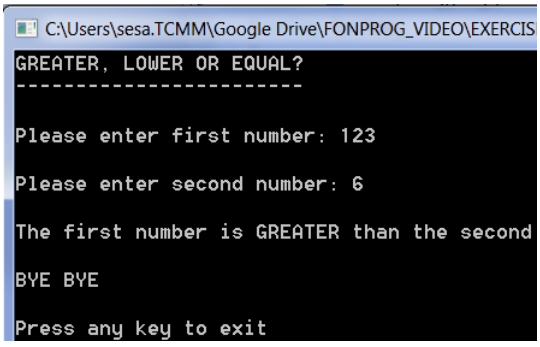


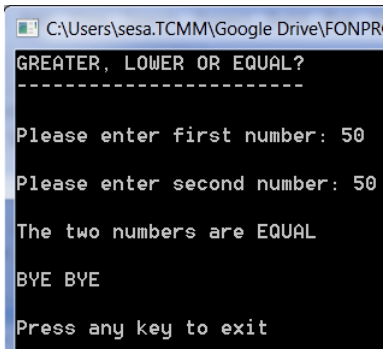


If they are the same, that's not a problem!

2. Now write a program that gets two integer numbers and tells if the first number is greater, lower o equal than the second. At the end the program always says BYE BYE.







See solution of previous exercises on next pages

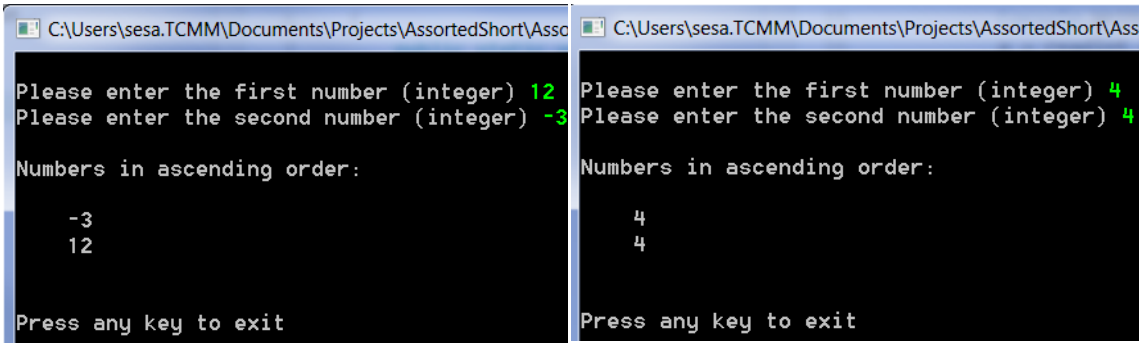
```
// EXERCISE 0a SOLUTION
static void Main(string[] args)
{
    int age;

    Console.WriteLine("TO BUY OR NOT TO BUY WHISKY");
    Console.WriteLine("--------------------------");
    Console.WriteLine();

    Console.Write("Please, tell me your age: ");
    age = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine();

    Console.WriteLine($"Being {age} years old...");
    if (age >= 18)
    {
        Console.WriteLine("    YOU ARE ALLOWED TO BUY WHISKY");
    }
    else
    {
        Console.WriteLine("    YOU ARE NOT ALLOWED TO BUY WHISKY");
        Console.WriteLine("            ---");
    }

    Console.WriteLine();
    Console.Write("Press any key to exit ");
    Console.ReadKey();
}
```

```
// EXERCISE 0b SOLUTION
static void Main(string[] args)
{
    double firstNumber, secondNumber;

    Console.WriteLine("NUMBER COMPARISON");
    Console.WriteLine("-----------------");
    Console.WriteLine();

    Console.Write("Please enter a number: ");
    firstNumber = Convert.ToDouble(Console.ReadLine());
    Console.Write("Please enter another number: ");
    secondNumber = Convert.ToDouble(Console.ReadLine());

    Console.WriteLine();
    Console.Write($"The numbers {firstNumber} and {secondNumber} are ");

    if (firstNumber == secondNumber)
    {
        Console.WriteLine("EQUAL");
    }
    else
    {
        Console.WriteLine("DIFFERENT");
    }

    Console.WriteLine();
    Console.Write("Press any key to exit ");
    Console.ReadKey();
}
```

```
// EXERCISE 1
public static void Main (string[] args)
{
    int firstNumber, secondNumber;

    Console.WriteLine ("WHICH NUMBER IS GREATER?");
    Console.WriteLine ("-----------------------");
    Console.WriteLine ();

    Console.Write ("Please enter first number: ");
    firstNumber = Convert.ToInt32 (Console.ReadLine ());
    Console.WriteLine ();
    Console.Write ("Please enter second number: ");
    secondNumber = Convert.ToInt32 (Console.ReadLine ());
    Console.WriteLine ();

    if (firstNumber > secondNumber) {
        Console.WriteLine ("The first number IS greater than the second");
    } else {
        Console.WriteLine ("The first number IS NOT  greater than the second");
    }

    Console.WriteLine ();
    Console.WriteLine ("BYE BYE");
    Console.WriteLine ();
    Console.Write ("Press any key to exit");
    Console.ReadKey (true);
}
```

```
// EXERCISE 2
public static void Main (string[] args)
{
    int firstNumber, secondNumber;

    Console.WriteLine ("GREATER, LOWER OR EQUAL?");
    Console.WriteLine ("-----------------------");
    Console.WriteLine ();

    Console.Write ("Please enter first number: ");
    firstNumber = Convert.ToInt32 (Console.ReadLine ());
    Console.WriteLine ();
    Console.Write ("Please enter second number: ");
    secondNumber = Convert.ToInt32 (Console.ReadLine ());
    Console.WriteLine ();

    if (firstNumber > secondNumber) {
        Console.WriteLine ("The first number is GREATER than the second");
    }
    else {
        if (firstNumber < secondNumber) {
            Console.WriteLine("The first number is LOWER than the second");
        }
        else {
            Console.WriteLine("The two numbers are EQUAL");
        }
    }

    Console.WriteLine ();
    Console.WriteLine ("BYE BYE");
    Console.WriteLine ();
    Console.Write ("Press any key to exit");
    Console.ReadKey (true);
}
```

3. Write a program that gets two integer numbers from its user and prints (writes) them in ascending order (if the first is lower than the second, first print it and then print the other; if the second is lower –or equal- than the first, print the second and then the first)



4. Same as before but instead of getting the numbers from its user, the program generates them randomly (in the interval [-10,10])

Remember that to generate random numbers, your program needs a random number generator, like …

```
Random alea = new Random (); // alea is a random number generator
```

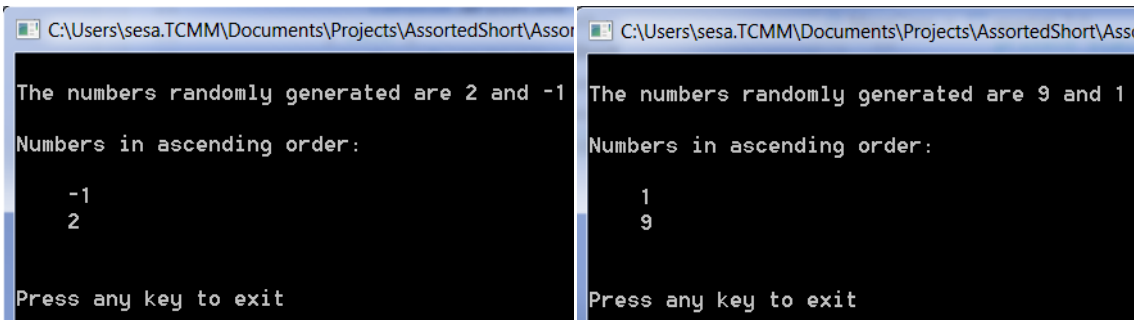
Then

```
alea.Next(α, β);
```

generates a random number in the interval [α, β)
(Notice that α is included but β is not included)

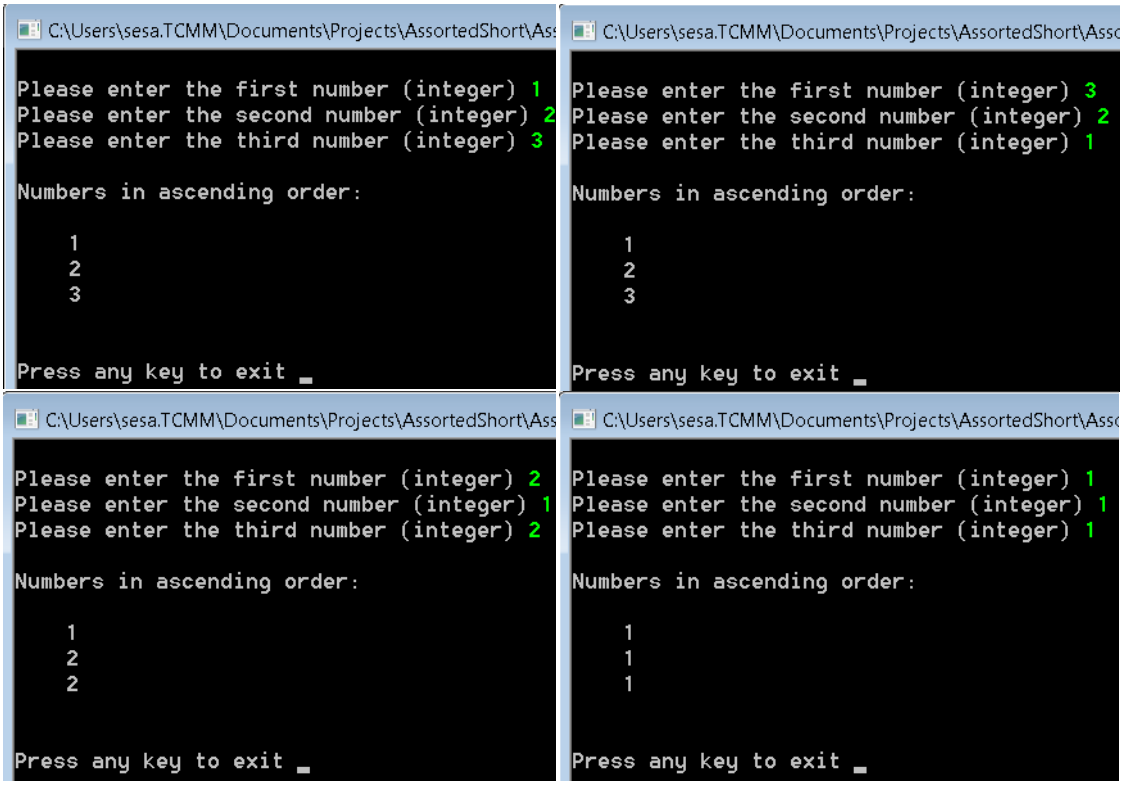
The generated number should be stored in a variable

```
int a;
a = alea.Next(-10, 11); // interval is [-10, 11) or [-10, 10]
```

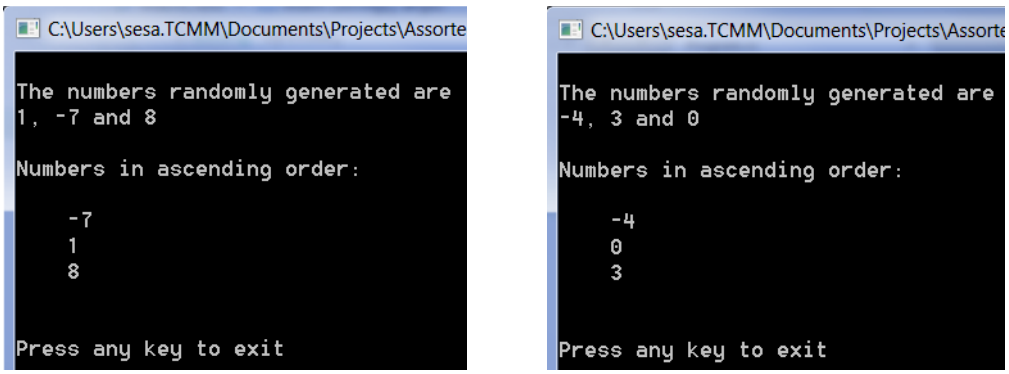


5. Write a program that gets three integer numbers and prints them in ascending order. Repetitions should not be a problem -do not worry about them, you do not have to do anything special- (see two last screen captures)

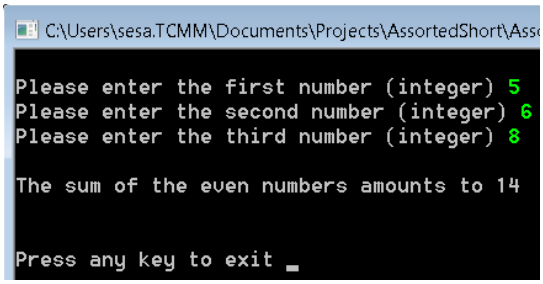
> **Beware:** for the novice programmer, this problem is not very easy…
> Use pencil & paper to analyse it. This may help you save time…

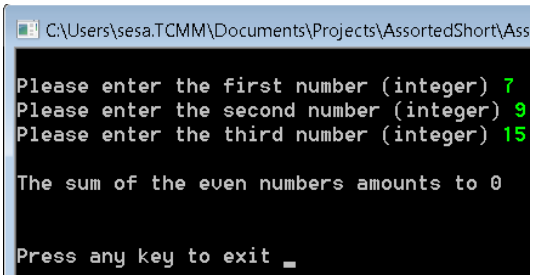


6. Same as before but generating the numbers randomly (in [-10, 10]).

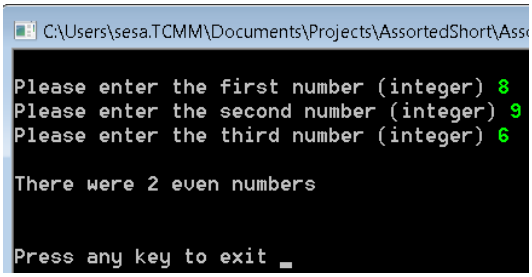


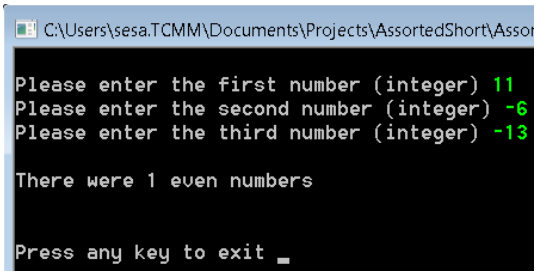
7. Write a program that gets three integer numbers and prints **the sum** of the even ones (if none is even then the sum is zero)





8. Write a program that gets three integer numbers and prints **how many** of them are even





Regarding exercises 7 and 8, notice that “to sum” and “to count” are not the same thing. In both cases are variable to hold the result is required. In the first case the variable acts as an **accumulator**. In the second case it acts as a **counter**. In both cases **it must be initialized to 0** (zero)

See solutions of exercises 3, 4, 5, 7 and 8 on next pages

```
// Exercise 3. Solution
public static void Main (string[] args)
{
    int a, b;

    Console.WriteLine ();
    Console.Write("Please enter the first number (integer) ");
    a = Convert.ToInt32 (Console.ReadLine());
    Console.Write ("Please enter the second number (integer) ");
    b = Convert.ToInt32 (Console.ReadLine());

    // if a is greater than b, first write b and than a
    // else write a and then b

    if (a > b) {
        Console.WriteLine ($"    {b} \n    {a}");
    } else {
        Console.WriteLine ($"    {a} \n    {b}");
    }

    // notice that if a and b are equal, the else branch
    // is executed. This is correct

    Console.WriteLine ("\n");
    Console.Write ("Press any key to exit ");
    Console.ReadKey ();
}
```

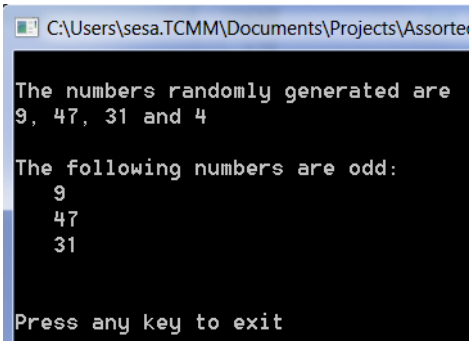
```
// Exercise 4. Solution
public static void Main (string[] args)
{
    Random alea = new Random (); // alea is a random number generator

    int a, b;

    a = alea.Next(-10, 11); // interval is [-10, 11) or [-10, 10]
    b = alea.Next (-10, 11);

    Console.WriteLine ();
    Console.WriteLine ($"The numbers randomly generated are {a} and {b}");
    Console.WriteLine ();
    Console.WriteLine ("Numbers in ascending order:\n");

    if (a > b) {
        Console.WriteLine ($"    {b} \n    {a}");
    } else {
        Console.WriteLine ($"    {b} \n    {a}");
    }

    Console.WriteLine ("\n");
    Console.Write ("Press any key to exit ");
    Console.ReadKey (true);
}
```

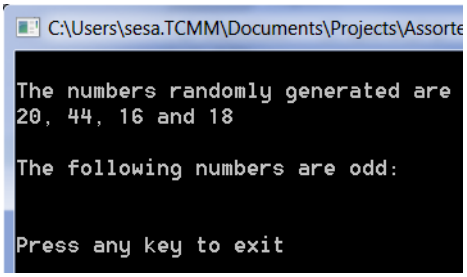
```
// Exercise 5. solution
public static void Main (string[] args)
{
    int a, b, c;

    ... //... irrelevant code not shown

    if (a <= b && a <= c) {
        // a is the lowest
        if (b < c) {
            Console.WriteLine ($"    {a} \n    {b}\n    {c}");
        } else {
            Console.WriteLine ($"    {a} \n    {c}\n    {b}");
        }
    } else if (b <= a && b <= c) {
        // b is the lowest
        if (a < c) {
            Console.WriteLine ($"    {b} \n    {a}\n    {c}");
        } else {
            Console.WriteLine ($"    {b} \n    {c}\n    {a}");
        }
    } else {
        // c is the lowest
        if (a < b) {
            Console.WriteLine ($"    {c} \n    {a}\n    {b}");
        } else {
            Console.WriteLine ($"    {c} \n    {b}\n    {a}");
        }
    }

    ... //... irrelevant code not shown
}
```

```
// EXERCISE 7
public static void Main (string[] args)
{
    int a, b, c;
    int sum;  // variable sum acts as an accumlator

    ... // ... means that some code is not being shown

    sum = 0;

    if (a % 2 == 0) {
        sum = sum + a;   // accumulate if even
    }

    if (b % 2 == 0) {
        sum = sum + b;  // accumulate if even
    }

    if (c % 2 == 0) {
        sum = sum + c;  // accumulate if even
    }

    // notice that the three if's are independent

    Console.WriteLine ($"The sum of the even numbers amounts to {sum}");

    ...
}
```

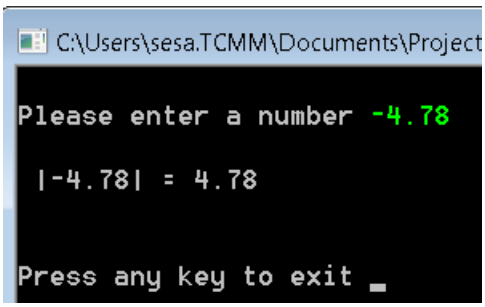
```
// EXERCISE 8
public static void Main(string[] args)
{
    int a, b, c;
    int howMany;  // variable howMany acts as a counter

    ...

    howMany = 0;

    if (a % 2 == 0){
        howMany = howMany + 1; // count if even
    }
    if (b % 2 == 0) {
        howMany = howMany + 1; // count if even
    }
    if (c % 2 == 0) {
        howMany = howMany + 1; // count if even
    }

    // notice that the three if's are independent

    Console.WriteLine ($"There were {howMany} even numbers");

    ...
}
```

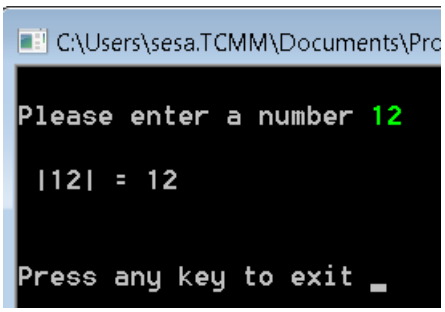
9. Write a program that generates four integer numbers (randomly, in the interval [0, 50]) prints them and then prints (again) the odd ones





10. Write a program that gets a floating point number from its user and prints its absolute value. (Do not use Math.Abs. Instead, check whether the given number is positive or not and act accordingly).





Several solutions for exercise 10 on next page

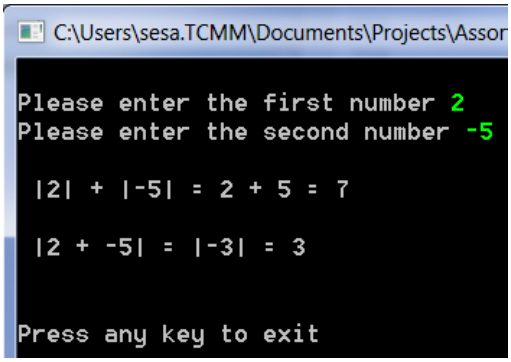
If your solution uses more than one variable, try to devise one that only requires one variable

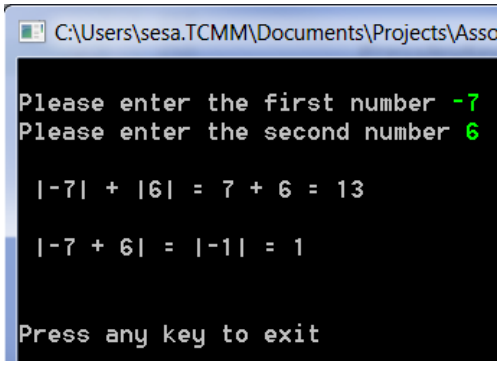
```
// EXERCISE 10, solution 1
public static void Main (string[] args)
{
    double v;
    double valAbs;

    Console.WriteLine ();
    Console.Write ("Please enter a number ");
    Console.ForegroundColor = ConsoleColor.Green;
    v = Convert.ToDouble (Console.ReadLine ());
    Console.ResetColor ();
    Console.WriteLine ();

    if (v < 0)
    {
        valAbs = v * -1.0;
        // valAbs = -v;  would also be correct
    }
    else
    {
        valAbs = v;
    }

    Console.WriteLine ($" |{v}| = {valAbs}");

    ...
}
```

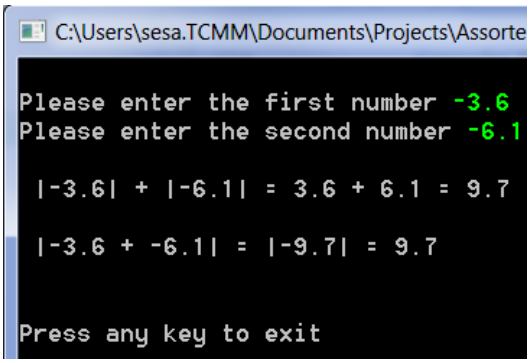
```
// EXERCISE 10, solution 2
public static void Main (string[] args)
{
    double v;
    double valAbs;

    Console.WriteLine ();
    Console.Write ("Please enter a number ");
    Console.ForegroundColor = ConsoleColor.Green;
    v = Convert.ToDouble (Console.ReadLine ());
    Console.ResetColor ();
    Console.WriteLine ();

    valAbs = v;
    if (valAbs < 0) {
        valAbs = -valAbs;
        // equivalent to valAbs = -1*valAbas
    }

    Console.WriteLine ($" |{v}| = {valAbs}");

    ...
}
```

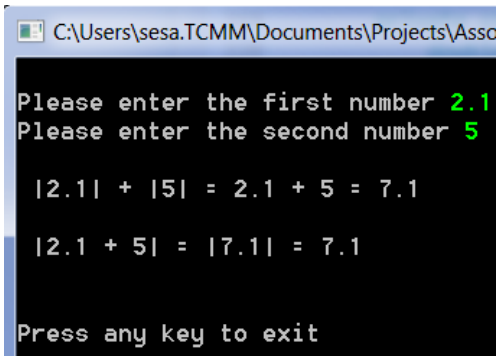
11. Write a program that gets two floating point numbers from its user and prints
    a) the sum of their absolute values
    b) The absolute value of their sum

    (Do not use Math.abs)

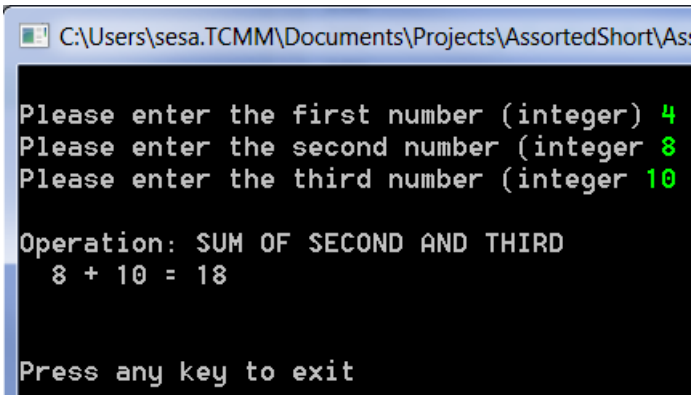


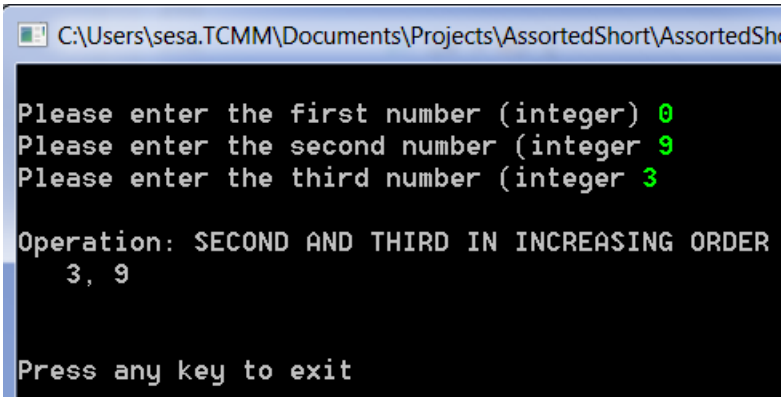


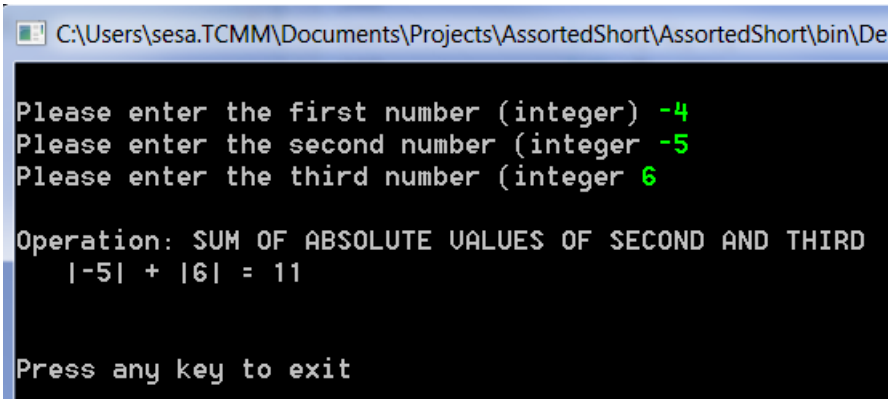




12. Write a program that gets three integer numbers from its user and then behaves as follows: if the first number is strictly positive (> 0) then the program computes and writes the sum of the second and the third; if it is zero then it writes the second and the third in increasing order; and if the first is negative it writes the sum of the absolute values of the second and the third.







Next page shows several schemas that lead to correct solutions

Schema 1: an if-else inside another if-else…

```
if (a>0)
{
    // sum of second and third (b+c)
}
else
{
    if (a==0)
    {
        // second and third in increasing order
    }
    else
    {
        // sum of absolute values of second and third
        // |b| + |c|
    }
}
```

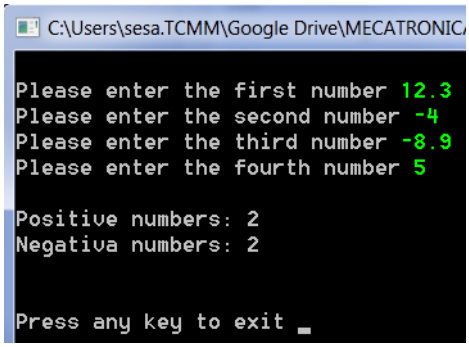
Schema 2: an if-else Ladder

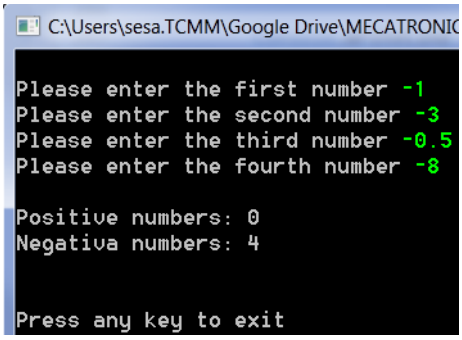
```
if (a>0)
{
    // sum of second and thirs (b+c)
}
else if (a==0)
{
    // second and third in increasing order
}
else
{
    // if here a<0
    // sum of absolute values of second and third
    // |b|+|c|
}
```

Schema 3: three dijoint ifs (no elses)

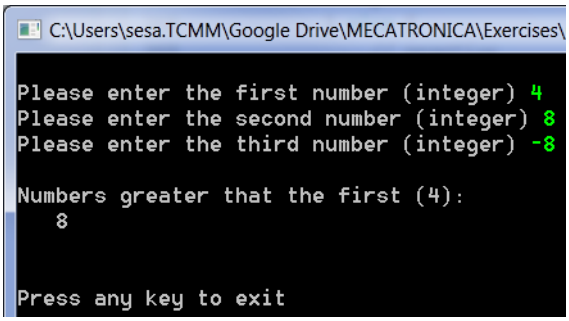
```
if (a>0)
{
    // sum of second and third (b+c)
}

if (a==0)
{
    // second and third in increasing order
}

if (a<0)
{
    // sum of absolute values of second and third
    // |b| + |c|
}
```

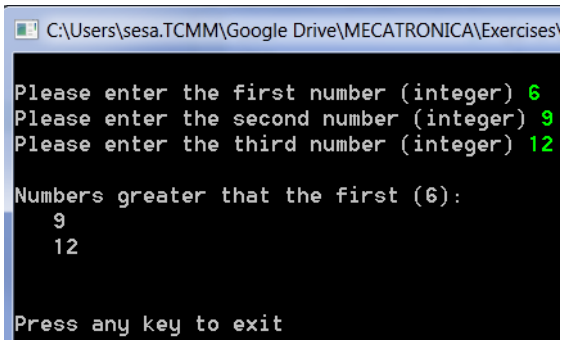
13. Write a program that gets four floating point numbers and counts (prints) how many of them are negative and how many are positive (consider zero a positive number). Would an if-else ladder be a good idea for this problem?

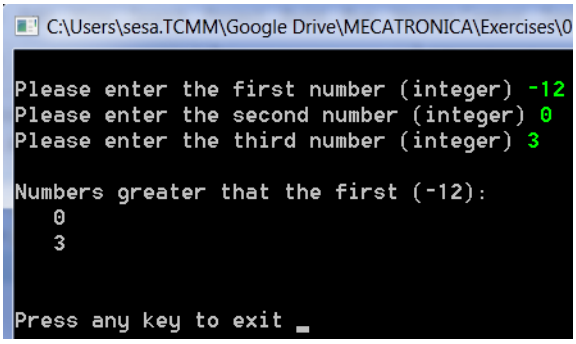


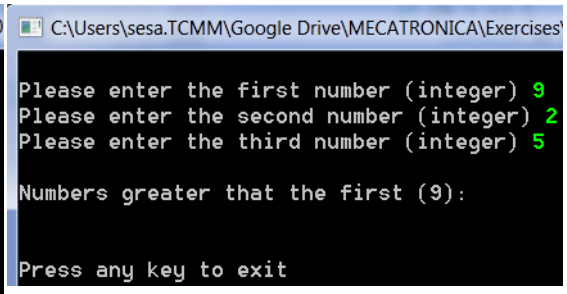


14. Write a program that gets three integer numbers and prints all the numbers that are greater than the first.

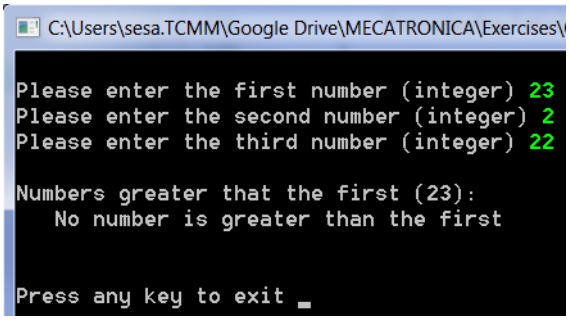


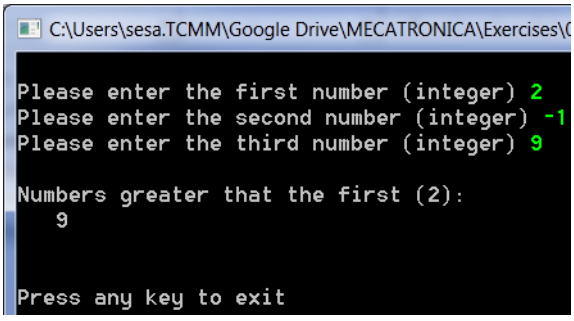




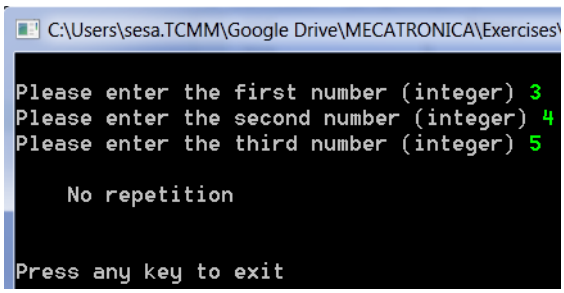


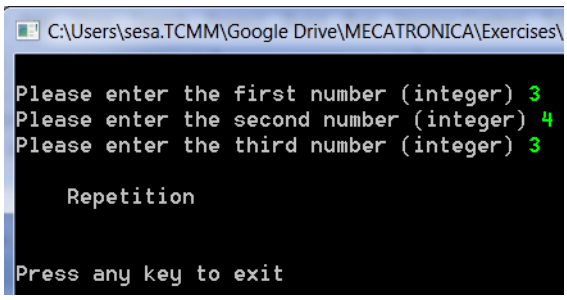
15. Same as before but if none is greater than the first the program prints ***No number is greater than the first*** instead of just not printing any number.

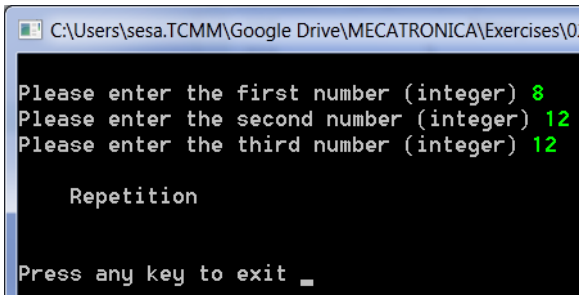


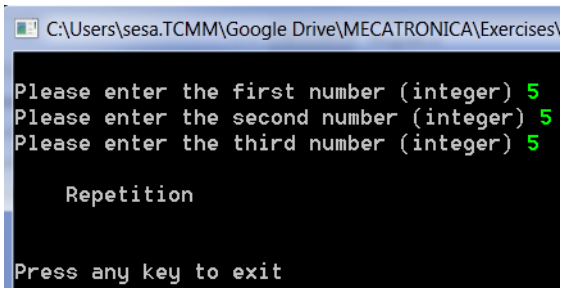


16. Write a program that gets three integer numbers and prints ***repetition*** or ***no repetition*** depending on whether any number appears more than once or not. Notice that the message is printed only once

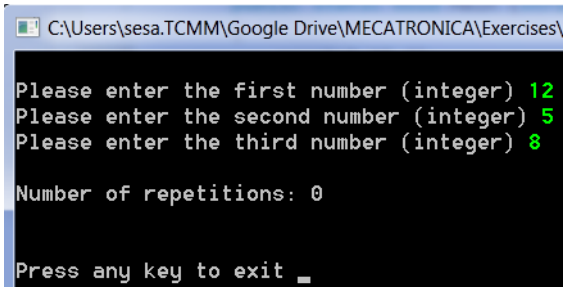


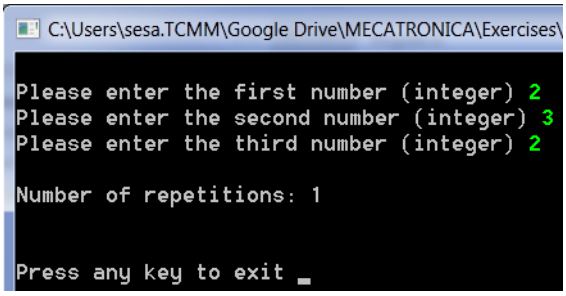


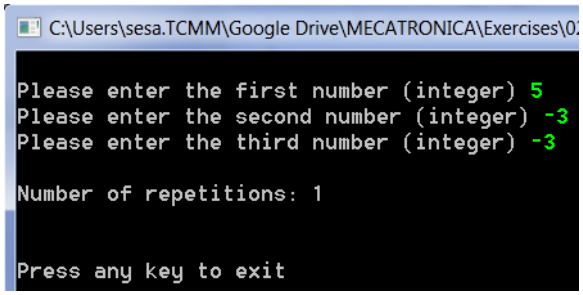


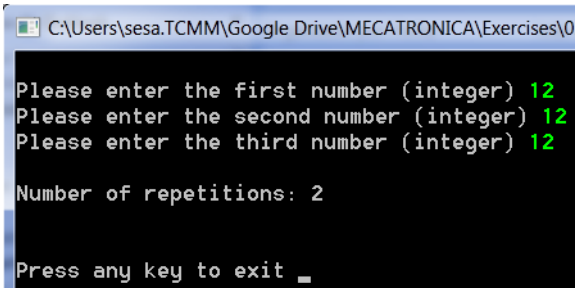


17. Write a program that gets three integer numbers and prints the number of repetitions (0, no repetition occurs, 1 meaning that two of the three numbers are equal or 2 when the three numbers are the same)

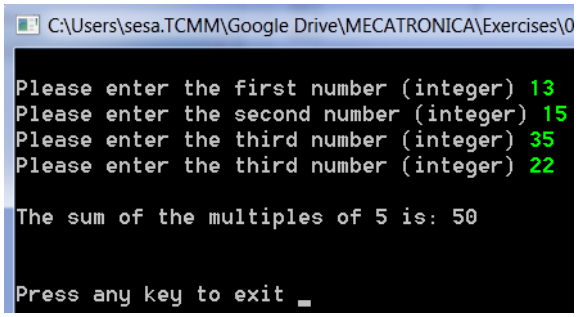


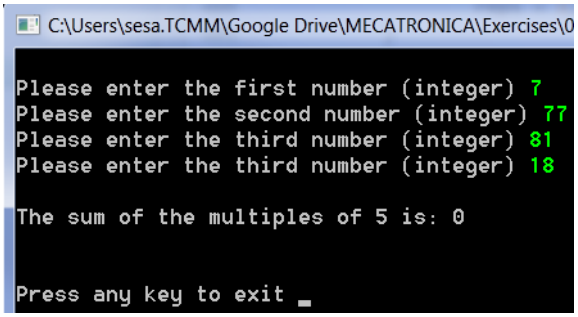


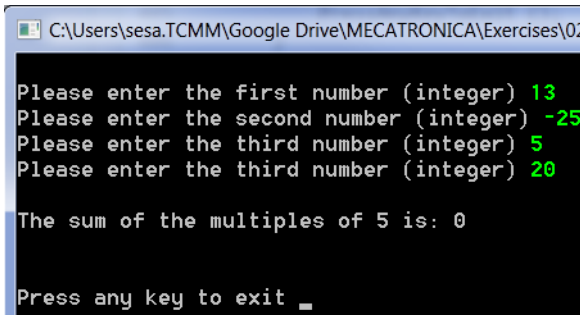


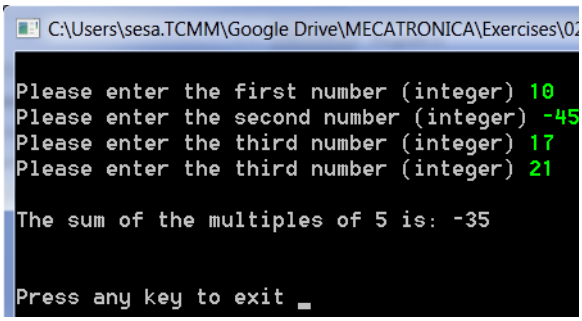


18. Write a program that gets four integer numbers and prints the sum of the multiples of 5. If no number is multiple of 5 the result must be 0

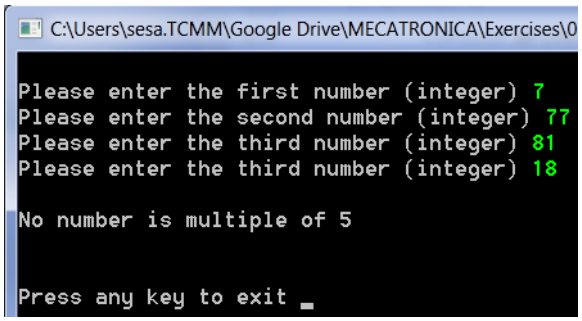


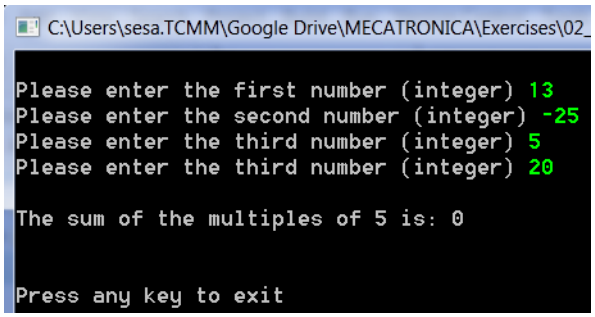




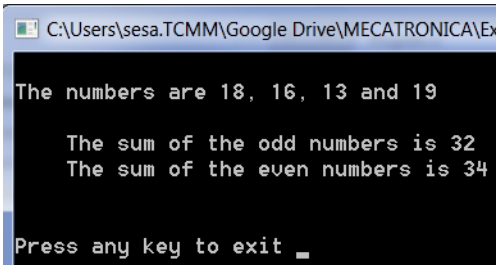


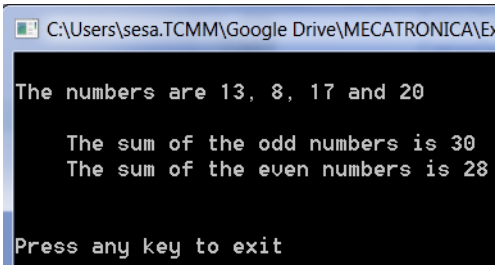
19. Same as before but now if no number is multiple of 5, the program prints ***No number is multiple of five*** instead of printing ***0***.

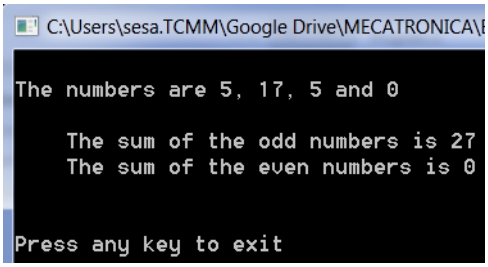


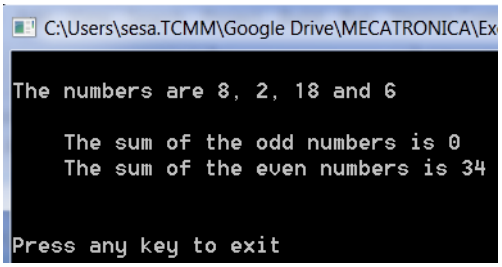


20. Write a program that randomly generates four integer numbers in the interval [0, 20], prints them and prints the sum of the odd ones and the sum of the even ones. If no number is odd, the sum is 0; if no number is even, the sum is also 0

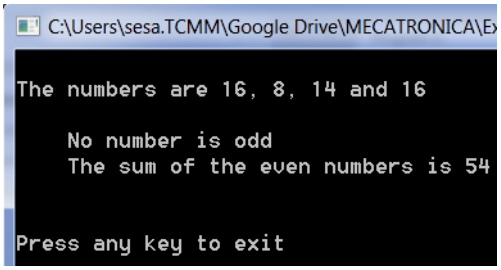


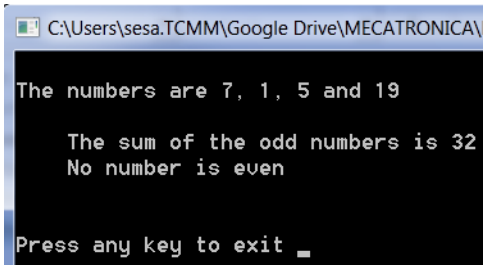






21. Same as before but if no number is odd the program prints ***No number is odd*** and if no number is even the program prints ***No number is even***.





Next pages show solutions for exercises 13, 14, 15, 16, 17, 18 and 21

```
// EXERCISE 13 SOLUTION
public static void Main (string[] args)
{
    double a, b, c, d;
    int positive=0, negative = 0;

    ...

    if (a >= 0) { positive++;}
    else { negative++;}

    if (b >= 0) { positive++; }
    else { negative++; }

    if (c >= 0) { positive++; }
    else { negative++; }

    if (d >= 0) { positive++; }
    else { negative++; }

    Console.WriteLine ($"Positive numbers: {positive}");
    Console.WriteLine ($"Negative numbers: {negative}");

    ...
}
```

```
// Exercise 14. Solution
public static void Main (string[] args)
{
    int a, b, c;

    ...

    Console.WriteLine ($"Numbers greater that the first ({a}):");

    if (b > a) {
        Console.WriteLine ($"   {b}");
    }

    if (c > a) {
        Console.WriteLine ($"   {c}");
    }

    /* notice that the ifs are independent since what happens with
     * the first number does not affect what will happen with the second
     */

    ...
}
```

```
// Exercise 15. Solution
public static void Main (string[] args)
{
    int a, b, c;
    int greater = 0; // count how many are greater than the first

    ...

    Console.WriteLine ($"Numbers greater that the first ({a}):");
    if (b > a) {
        Console.WriteLine ($"   {b}");
        greater++;
    }
    if (c > a) {
        Console.WriteLine ($"   {c}");
        greater++;
    }

    if (greater == 0) {
        Console.WriteLine ("   No number is greater than the first");
    }

    ...
}
```

```
// Exercise 16. Solution
public static void Main (string[] args)
{
    int a, b, c;

    ...

    if (a == b || a == c || b == c) {
        Console.WriteLine ("    Repetition");
    } else {
        Console.WriteLine ("    No repetition");
    }

    /* Notice that there's no need to check
     * b==a or c==a or c==b ...
     */

    ...
}
```

```
// Exercise 17. Solution
public static void Main (string[] args)
{
    int a, b, c;
    int reps = 0; // this variable will count the number of repetitions

    ...


    if ((a == b) && (b == c)) { reps = 2; }
    else {
        if (a == b || a == c || b == c) { reps = 1; }
    }

    Console.WriteLine ($"Number of repetitions: {reps}");

    ...
}
```

```
// Exercise 18. Solution
public static void Main (string[] args)
{
    int a, b, c,d;
    int sum = 0;

    ...

    if (a % 5 == 0) {
        sum = sum + a;
    }
    if (b % 5 == 0) {
        sum = sum + b;
    }
    if (c % 5 == 0) {
        sum = sum + c;
    }
    if (d % 5 == 0) {
        sum = sum + d;
    }

    // notice that the four ifs are independent from each other
    // since what happens with any one of the numbers has no effect
    // on what will happen with any other

    Console.WriteLine ($"The sum of the multiples of 5 is: {sum}");

    ...
}
```

```
// exercise 21. Solution
public static void Main (string[] args)
{
    int a, b, c,d;
    int sumOdd, sumEven; // accumulators for odd and even
    int nOdd, nEven;  // counters for odd and even

    Random alea = new Random ();

    a = alea.Next (0, 21);
    b = alea.Next (0, 21);
    c = alea.Next (0, 21);
    d = alea.Next (0, 21);

    Console.WriteLine ();
    Console.WriteLine ($"The numbers are {a}, {b}, {c} and {d}");
    Console.WriteLine ();

    sumOdd = 0;
    sumEven = 0;
    nOdd = 0;
    nEven = 0;

    if (a % 2 == 0) { sumEven = sumEven + a; nEven++;}
    else { sumOdd = sumOdd + a; nOdd++;}

    if (b % 2 == 0) { sumEven = sumEven + b; nEven++;}
    else { sumOdd = sumOdd + b; nOdd++;}

    ... // ... (same with c and d)

    if (nOdd != 0) {
        Console.WriteLine ($"    The sum of the odd numbers is {sumOdd}");
    } else {
        Console.WriteLine ("    No number is odd");
    }

    if (nEven != 0) {
        Console.WriteLine ($"    The sum of the even numbers is {sumEven}");
    } else {
        Console.WriteLine ("    No number is even");
    }

    ...
}
```

BOOLEAN EXPRESSIONS (1)

Dogs, cats and rats

Consider the following fragment of code:

```
int dogs, cats, rats;

Console.Write ("How many dogs are there? ");
dogs = Convert.ToInt32 (Console.ReadLine());

Console.Write ("and how many cats? ");
cats = Convert.ToInt32 (Console.ReadLine());

Console.Write ("and, finally, how many rats? ");
rats = Convert.ToInt32 (Console.ReadLine());

if (?) {
    Console.WriteLine ("yep, that's true!!!");
}
else {
    Console.WriteLine ("nope, that's false");
}
```

Give the conditions (bool expressions) for the following assertions

a) The number of rats is three times the number of cats

b) The number of rats is at least three times the number of cats

c) There are more rats than cats and more cats than dogs

d) It is not true that there are more rats than cats and more cats than dogs

e) There is the same number of dogs, cats and rats

f) There is an odd number of rats and an even number of cats

g) None of the three numbers is odd

h) At least one of the three numbers is even

i) Exactly one of the three numbers is even

j) The three numbers are all different

k) At least two of the three numbers are equal

l) Two, and only two, of the three numbers are equal

m) All three numbers have the same parity (all three numbers are odd or all three numbers are even)

SOLUTIONS

a) The number of rats is three times the number of cats

```
rats == 3*cats
```

b) The number of rats is at least three times the number of cats

```
rats >= 3*cats
```

c) There are more rats than cats and more cats than dogs

```
(rats > cats) && (cats > dogs)
```

d) It is not true that there are more rats than cats and more cats than dogs

```
!((rats > cats) && (cats > dogs))
```
also possible…

```
((rats > cats) && (cats > dogs)) == false
```
also possible…

```
(rats <= cats) || (cats <= dogs)
```

e) There is the same number of dogs, cats and rats

```
(dogs==cats)&&(cats==rats)
```

f) There is an odd number of rats and an even number of cats

```
(rats%2==1) && (cats%2==0)
```

g) None of the three numbers is odd

```
(rats%2!=1) && (cats%2!=1) && (dogs%2!=1)
```
also possible…

```
(rats%2==0) && (cats%2==0) && (dogs%2==0)
```

h) At least one of the three numbers is even

```
(rats%2==0) || (cats%2==0) || (dogs%2==0)
```

i) Exactly one of the three numbers is even

```
(rats%2==0 && cats%2!=0 && dogs%2!=0)
||
(cats%2==0 && rats%2!=0 && dogs%2!=0)
||
(dogs%2==0 && rats%2!=0 && cats%2!=0)
```

j) The three numbers are all different

```
dogs!=cats && cats!=rats && dogs!=rats
```

k) At least two of the three numbers are equal

```
!(dogs!=cats && cats!=rats && dogs!=rats)
dogs==cats || cats==rats || dogs==rats
```

l) Two, and only two, of the three numbers are equal

```
((dogs==cats) && (dogs!=rats))
||
((cats==rats) && (cats!=dogs))
||
((dogs==rats) && (cats!=rats))
```

m) All three numbers have the same parity (all three numbers are odd or all three numbers are even)

```
dogs%2==0 && cats%2==0 && rats%2==0
||
dogs%2==1 && cats%2==1 && rats%2==1
```

BOOLEAN EXPRESSIONS (2)

With...

```
int friends, foes;

friends = ...
foes = ...
```

Write conditions for:

a) There are between 200 and 300 foes (both included)

b) There are more than 200 friends or the number of foes does not exceed 100.

c) The number of foes is in the interval [10, 50)

d) The number of friends is not in the interval [5, 15]

With...

```
double mark1, mark2, mark3, mark4;

mark1 = ...
mark2 = ...
mark3 = ...
mark4 = ...
```

Write the conditions that express:

a) All marks are five or more

b) At least one mark does not reach five

With ...

```
int age;
char nutrition; // 'v' means vegetarian
                // any other character different from 'v' means carnivore

age = ...
nutrition = ...
```

Write conditions for

a) Vegetarian of over thirty years (more than thirty)

b) Carnivore or underaged (cat: carnívor o menor d'edat)

With ...

```
int iq;    // Intelligence Quotient (IQ)
char shy;  // 'y' and 'Y' mean the person is shy
           // any other value means not shy (extroverted)

iq = ...
shy = ...
```

Write the conditions that express

a) Extroverted person with an IQ in the range [80, 115)

b) Shy person with an IQ higher than 130

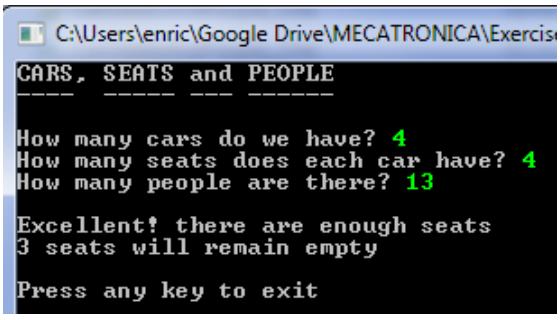
OTHER EXERCISES

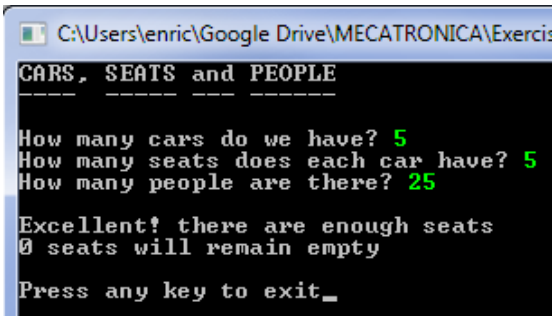
Each exercise is followed by its solution

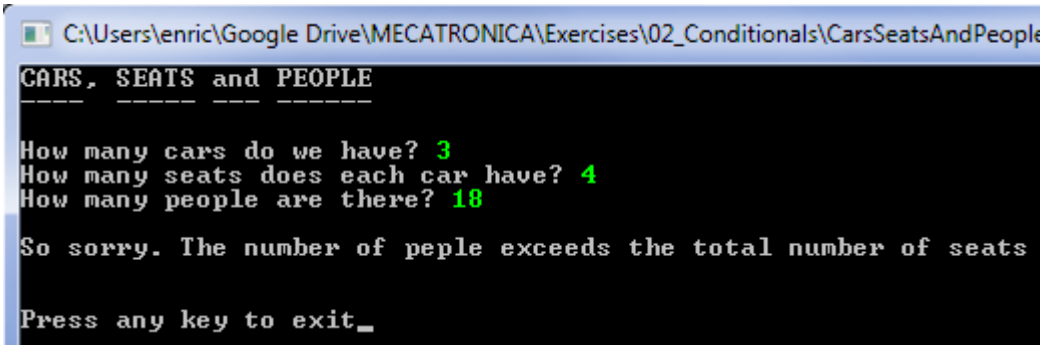
Exercise “Cars, seats and people”

Write (complete) a program that given a number of cars, the number of seats in every car (all cars have the same number of seats) and a number of people, determines whether there are enough seats to transport every one or not. If the answer is positive, the program also computes the number of seats that will remain empty. If the answer is negative the program just “regrets” the situation.

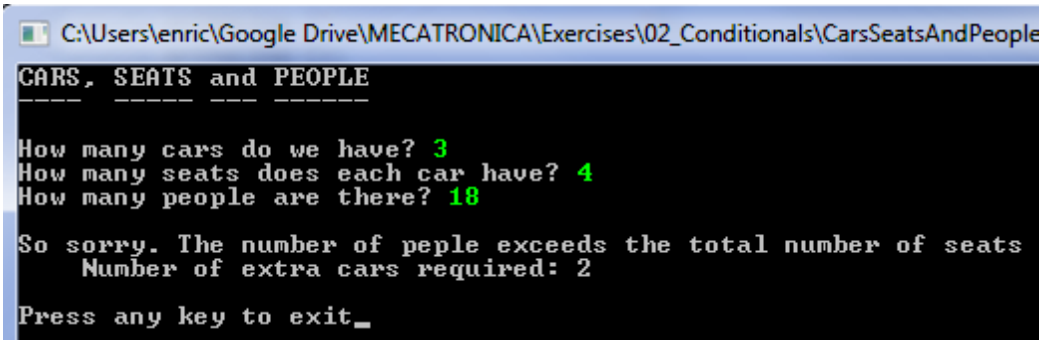
```
using System;

namespace CarsSeatsAndPeople_One
{
    class MainClass
    {
        public static void Main (string[] args)
        {
            int numCars, seatsPerCar, people;

            /* add more variables if needed */


            Console.WriteLine ("CARS, SEATS and PEOPLE");
            Console.WriteLine ("----  ----- --- ------");
            Console.WriteLine ();
            Console.Write ("How many cars do we have? ");
            numCars = Convert.ToInt32(Console.ReadLine());
            Console.Write ("How many seats does each car have? ");
            seatsPerCar = Convert.ToInt32(Console.ReadLine());
            Console.Write ("How many people are there? ");
            people = Convert.ToInt32(Console.ReadLine());

            Console.WriteLine ();

            /* COMPLETE */









            Console.SetCursorPosition (0, 10);
            Console.Write ("Press any key to exit");
            Console.ReadKey ();
        }
    }
}
```

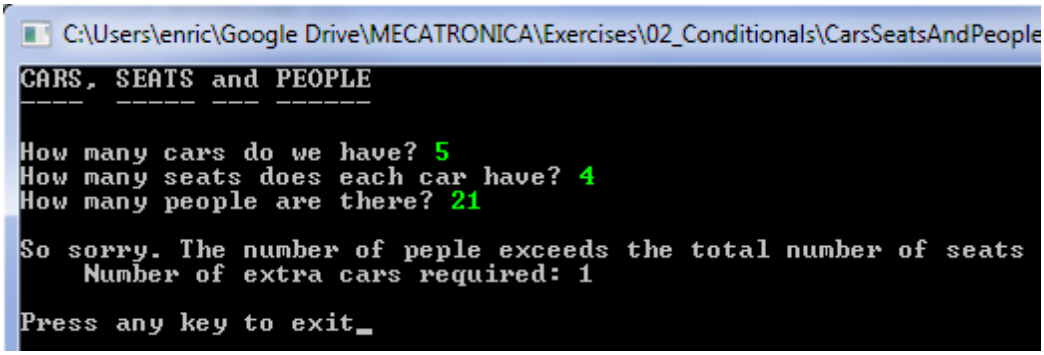


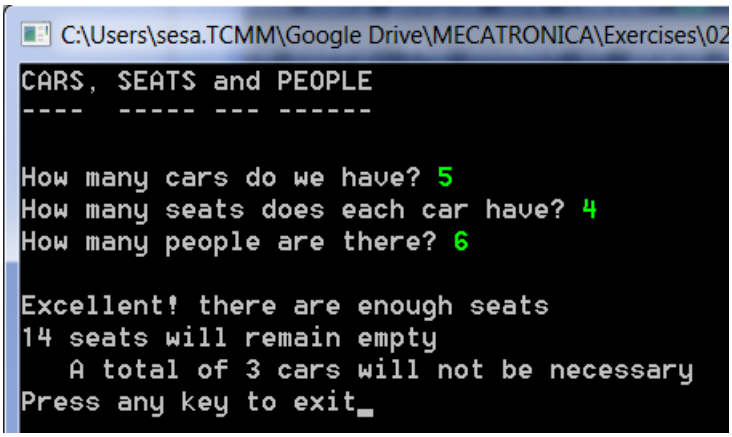


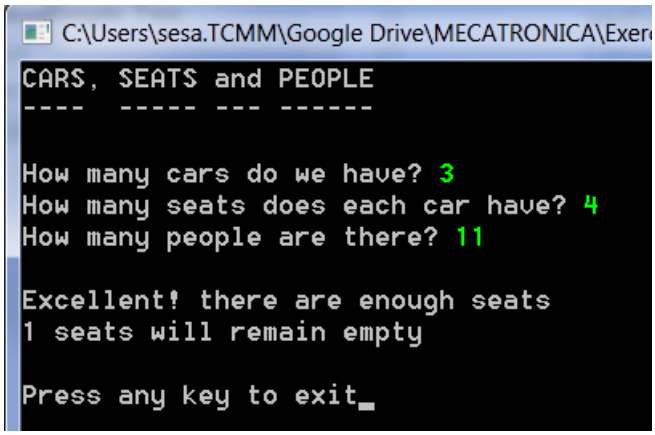


Improve the program. Now, if there are not enough seats, the program also computes the number of extra cars required to transport every one; and if the number of seats is enough the program computes how many seats will remain empty and also how many cars will not be necessary.






C:\Users\sesa.TCMM\Google Drive\MECATRONICA\Exercises\02
CARS, SEATS and PEOPLE
---- ----- --- ------
How many cars do we have? 5
How many seats does each car have? 4
How many people are there? 6
Excellent! there are enough seats
14 seats will remain empty
A total of 3 cars will not be necessary
Press any key to exit_


C:\Users\sesa.TCMM\Google Drive\MECATRONICA\Exer
CARS, SEATS and PEOPLE
---- ----- --- ------
How many cars do we have? 3
How many seats does each car have? 4
How many people are there? 11
Excellent! there are enough seats
1 seats will remain empty
Press any key to exit_

```
// Cars, seats and People. Solution
public static void Main (string[] args)
{
    int numCars, seatsPerCar, people;
    int totalSeats;
    int emptySeats;

    Console.WriteLine ("CARS, SEATS and PEOPLE");
    Console.WriteLine ("----  ----- --- ------");
    Console.WriteLine ();

    ...

    Console.WriteLine ();

    totalSeats = numCars * seatsPerCar;

    if (totalSeats < people) {
        Console.WriteLine ("So sorry. The number of peple exceeds the total number of seats");
    }
    else {
        emptySeats = totalSeats - people;
        Console.WriteLine ("Excellent! there are enough seats");
        Console.WriteLine ($"{emptySeats} seats will remain empty");
    }

    ...
}
```

```
// Cars, seats and People. Extended version. Solution
public static void Main_TWO (string[] args)
{
    int numCars, seatsPerCar, people;
    int totalSeats;
    int emptySeats;
    int emptyCars, extraCars;

    ...

    totalSeats = numCars * seatsPerCar;

    if (totalSeats < people) {
        Console.WriteLine ("So sorry. The number of peple exceeds the total number of seats");
        extraCars = (people - totalSeats) / seatsPerCar;
        if ((people - totalSeats) % seatsPerCar != 0) {
            extraCars = extraCars + 1;
        }
        Console.WriteLine ($"    Number of extra cars required: {extraCars}");
    }
    else {
        emptySeats = totalSeats - people;
        emptyCars = emptySeats / seatsPerCar;
        Console.WriteLine ("Excellent! there are enough seats");
        Console.WriteLine ($"{emptySeats} seats will remain empty");
        if (emptyCars > 0) {
            Console.WriteLine ($"   A total of {emptyCars} cars will not be necessary");
        }
    }

    ...
}
```

Exercise "Harry Potter's grading system"

Write the **if-else ladder** structure of a program in which the user enters a mark (in [0,10]) and then the mark is expressed using Harry Potter's O.W.L.s grading system:

[0,2) Troll (T)
[2, 4) Dreadful (D)
[4, 5) Poor (P)
[5, 8] Acceptable (A) - beware fully closed interval-
(8, 10) Exceeds Expectations (E)
10 Outstanding (O)

```
static void Main(string[] args)
{
    double mark;

    Console.WriteLine("HARRY POTTER's O.W.L.s GRADING SYSTEM");
    Console.WriteLine("-------------------------------------");
    Console.WriteLine();

    Console.Write("Hi muggle! Enter your [0, 10] mark ");
    Console.ForegroundColor = ConsoleColor.Yellow;
    mark = Convert.ToDouble(Console.ReadLine());
    Console.ResetColor();
    Console.WriteLine();

    Console.WriteLine($"In the O.W.L.s Grading System {mark} is ");

    /* IF-ELSE LADDER GOES HERE */

    Console.ResetColor();
    Console.SetCursorPosition(0, Console.WindowHeight - 1);
    Console.Write("press any key to exit");
    Console.ReadKey();
}
```

```
HARRY POTTER's O.W.L.s GRADING SYSTEM
-------------------------------------

Hi muggle! Enter your [0, 10] mark 1.5

In the O.W.L.s Grading System 1.5 is
        Troll (T)
```

```
HARRY POTTER's O.W.L.s GRADING SYSTEM
-------------------------------------

Hi muggle! Enter your [0, 10] mark 2

In the O.W.L.s Grading System 2 is
        Dreadful (D)
```

```
HARRY POTTER's O.W.L.s GRADING SYSTEM
-------------------------------------

Hi muggle! Enter your [0, 10] mark 4

In the O.W.L.s Grading System 4 is
        Poor (P)
```

```
HARRY POTTER's O.W.L.s GRADING SYSTEM
-------------------------------------

Hi muggle! Enter your [0, 10] mark 5

In the O.W.L.s Grading System 5 is
        Acceptable (A)
```

```
HARRY POTTER's O.W.L.s GRADING SYSTEM
-------------------------------------

Hi muggle! Enter your [0, 10] mark 8

In the O.W.L.s Grading System 8 is
        Acceptable (A)
```

```
HARRY POTTER's O.W.L.s GRADING SYSTEM
-------------------------------------

Hi muggle! Enter your [0, 10] mark 8.001

In the O.W.L.s Grading System 8.001 is
        Exceeds expectations (E)
```

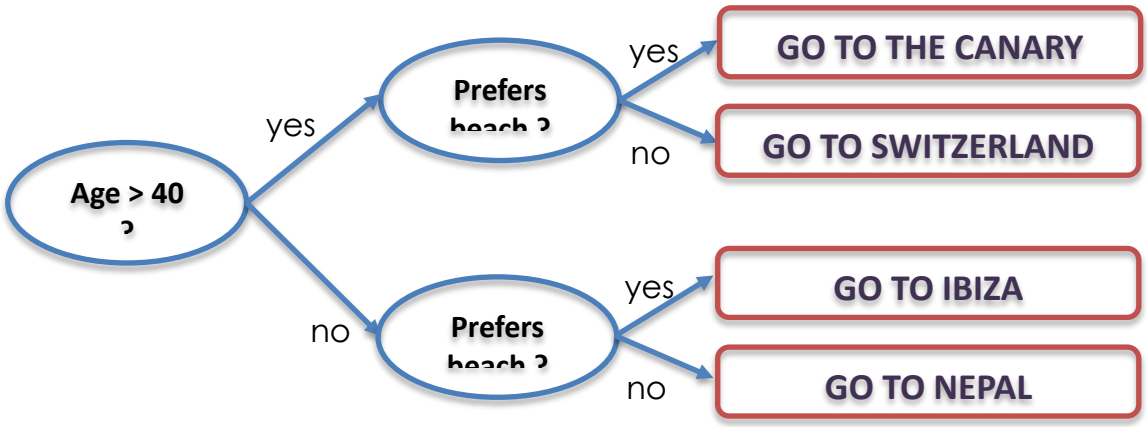
```
HARRY POTTER's O.W.L.s GRADING SYSTEM
-------------------------------------

Hi muggle! Enter your [0, 10] mark 9.9

In the O.W.L.s Grading System 9.9 is
        Exceeds expectations (E)
```

```
HARRY POTTER's O.W.L.s GRADING SYSTEM
-------------------------------------

Hi muggle! Enter your [0, 10] mark 10

In the O.W.L.s Grading System 10 is
        Outstanding (O)
```

```
// Harry Potter's Grading System. Solution
static void Main(string[] args)
{
    double mark;

    ...

    Console.WriteLine("In the O.W.L.s Grading System {0} is ", mark);

    if (mark < 0  || mark > 10)
    {
        Console.WriteLine("\t AN INCORRECT MARK. ALL MARKS MUST BE IN [0,10]");
    }
    else if (mark < 2) {
        Console.ForegroundColor = ConsoleColor.Red;
        Console.WriteLine("\t Troll (T)");
    }
    else if (mark < 4)
    {
        Console.ForegroundColor = ConsoleColor.Red;
        Console.WriteLine("\t Dreadful (D)");
    }
    else if (mark < 5)
    {
        Console.ForegroundColor = ConsoleColor.Red;
        Console.WriteLine("\t Poor (P)");
    }
    else if (mark <=8)
    {
        Console.ForegroundColor = ConsoleColor.Green;
        Console.WriteLine("\t Acceptable (A)");
    }
    else if (mark < 10)
    {
        Console.ForegroundColor = ConsoleColor.Green;
        Console.WriteLine("\t Exceeds expectations (E)");
    }
    else
    {
        Console.ForegroundColor = ConsoleColor.DarkGreen;
        Console.WriteLine("\t Outstanding (O)");
    }

    ...
}
```

Exercise "Holiday advisor"

Write (Complete) a C# program that gets a name, age and preference (beach or mountain) from its user and gives holiday advice following these rules




```
HOLIDAY ADVISOR
------- -------

Please, write your name ENRIC
Well ENRIC now tell me your age 23
Finally, tell me your preference...
...Do you prefer beach (b) or mountain (m)? b

My advice is that you go to IBIZA

Press any key to exit
```

```
HOLIDAY ADVISOR
------- -------

Please, write your name JOANA
Well JOANA now tell me your age 55
Finally, tell me your preference...
...Do you prefer beach (b) or mountain (m)? m

My advice is that you go to TO SWITZERLAND

Press any key to exit
```

```
// Holiday Advisor. Solution
public static void Main (string[] args)
{
    int age;
    String name;
    char preference;

    Console.WriteLine ("HOLIDAY ADVISOR");
    Console.WriteLine ("------- -------");
    Console.WriteLine ();

    Console.Write("Please, write your name ");
    name = Console.ReadLine ();
    Console.Write ($"Well {name} now tell me your age ");
    age = Convert.ToInt32 (Console.ReadLine());
    Console.WriteLine ("Finally, tell me your preference...");
    Console.Write ("...Do you prefer beach (b) or mountain (m)? ");
    preference = Convert.ToChar (Console.ReadLine());

    Console.WriteLine ();
    Console.Write ("My advice is that you go to ");


    if (age > 40) {
        if (preference == 'b') {
            Console.WriteLine ("THE CANARY ISLANDS");
        } else {
            Console.WriteLine ("TO SWITZERLAND");
        }
    } else {
        if (preference == 'b') {
            Console.WriteLine ("IBIZA");
        } else {
            Console.WriteLine ("NEPAL");
        }
    }

    ...
}
```

Exercise “Holiday Advisor #2”

Write the if-else ladder structure of yet another holiday advisor program based on gender and age.

males over forty but less than sixty => CANCUN
Males of sixty or more => CALELLA
Females of thirty or less => CUSCO
Females over sixty-five => CANNES
Other => CASABLANCA

```
static void Main(string[] args)
{

    int age;
    char gender;

    Console.WriteLine("YET ANOTHER HOLIDAY ADVISOR");
    Console.WriteLine("--------------------------");
    Console.WriteLine();

    Console.Write("Enter gender (m for male, f for female): ");
    gender = Convert.ToChar(Console.ReadLine());
    Console.Write("Enter age: ");
    age = Convert.ToInt32(Console.ReadLine());

    Console.WriteLine();
    Console.Write("This advisor suggests you travel to ");
    Console.ForegroundColor = ConsoleColor.Cyan;

    /* IF-ELSE LADDER COMES HERE */

    Console.ResetColor();
    Console.SetCursorPosition(0, Console.WindowHeight - 1);
    Console.Write("press any key to exit");
    Console.ReadKey();
}
```

```
YET ANOTHER HOLIDAY ADVISOR
--------------------------

Enter gender (m for male, f for female): m
Enter age: 40

This advisor suggests you travel to CASABLANCA
```

```
YET ANOTHER HOLIDAY ADVISOR
--------------------------

Enter gender (m for male, f for female): m
Enter age: 59

This advisor suggests you travel to CANCUN
```

```
YET ANOTHER HOLIDAY ADVISOR
--------------------------

Enter gender (m for male, f for female): m
Enter age: 60

This advisor suggests you travel to CALELLA
```

```
YET ANOTHER HOLIDAY ADVISOR
--------------------------

Enter gender (m for male, f for female): f
Enter age: 19

This advisor suggests you travel to CUSCO
```

```
YET ANOTHER HOLIDAY ADVISOR
--------------------------

Enter gender (m for male, f for female): f
Enter age: 30

This advisor suggests you travel to CUSCO
```

```
YET ANOTHER HOLIDAY ADVISOR
--------------------------

Enter gender (m for male, f for female): f
Enter age: 31

This advisor suggests you travel to CASABLANCA
```

```
YET ANOTHER HOLIDAY ADVISOR
--------------------------

Enter gender (m for male, f for female): f
Enter age: 65

This advisor suggests you travel to CASABLANCA
```

```
YET ANOTHER HOLIDAY ADVISOR
--------------------------

Enter gender (m for male, f for female): f
Enter age: 70

This advisor suggests you travel to CANNES
```

```
// Holiday Advisor 2. Solution
static void Main(string[] args)
{
    int age;
    char gender;

    ...

    Console.Write("Enter gender (m for male, f for female): ");
    gender = Convert.ToChar(Console.ReadLine());
    Console.Write("Enter age: ");
    age = Convert.ToInt32(Console.ReadLine());

    Console.WriteLine();
    Console.Write("This advisor suggests you travel to ");
    Console.ForegroundColor = ConsoleColor.Cyan;

    if (gender=='m' && age>40 && age<60)
    {
        Console.WriteLine("CANCUN");
    }
    else if (gender=='m' && age >= 60)
    {
        Console.WriteLine("CALELLA");
    }
    else if (gender=='f' && age<=30)
    {
        Console.WriteLine("CUSCO");
    }
    else if (gender=='f' && age > 65)
    {
        Console.WriteLine("CANNES");
    }
    else {
        Console.WriteLine("CASABLANCA");
    }

    ...
}
```

Exercise “Seasons”

Write a string-based switch structure that given the two-letter code of a season writes the names of the central months of that season

Two-letter codes are: SP, SU, AU and WI

Central months are:
Spring => April and May
Summer => July and August
Autumn => October and November
Winter => January and February

```
static void Main(string[] args)
{
    string seasonCode;

    Console.WriteLine("THE SEASONS");
    Console.WriteLine("----------");
    Console.WriteLine();

    Console.WriteLine("Enter season two-letter code");
    Console.Write("Spring(SP), Summer(SU), Autumn(AU), WINTER(WI) >> " );
    seasonCode = Console.ReadLine();
    Console.WriteLine();

    /* WRITE YOUR SWITCH HERE */

    Console.ResetColor();
    Console.SetCursorPosition(0, Console.WindowHeight - 1);
    Console.Write("Press any key to exit");
    Console.ReadKey();
}
```

```
THE SEASONS
----------

Enter season two-letter code
Spring(SP), Summer(SU), Autumn(AU), WINTER(WI) >> SP

APRIL and MAY are the central months of SPRING
```

```
THE SEASONS
----------

Enter season two-letter code
Spring(SP), Summer(SU), Autumn(AU), WINTER(WI) >> WI

JANUARY and FEBRUARY are the central months of WINTER
```

```
THE SEASONS
----------

Enter season two-letter code
Spring(SP), Summer(SU), Autumn(AU), WINTER(WI) >> AU

OCTOBER and NOVEMBER are the central months of AUTUMN
```

```
THE SEASONS
----------

Enter season two-letter code
Spring(SP), Summer(SU), Autumn(AU), WINTER(WI) >> SU

JULY and AUGUST are the central months of SUMMER
```

```
THE SEASONS
----------

Enter season two-letter code
Spring(SP), Summer(SU), Autumn(AU), WINTER(WI) >> KL

KL is not a correct season code
```

```
// Seasons. Solution
static void Main(string[] args)
{
    string seasonCode;

    Console.WriteLine("THE SEASONS");
    Console.WriteLine("----------");
    Console.WriteLine();

    Console.WriteLine("Enter season two-letter code");
    Console.Write("Spring(SP), Summer(SU), Autumn(AU), WINTER(WI) >> " );
    seasonCode = Console.ReadLine();
    Console.WriteLine();

    switch (seasonCode)
    {
        case "SP":
            Console.ForegroundColor = ConsoleColor.Green;
            Console.WriteLine("APRIL and MAY are the central months of SPRING");
            break;
        case "SU":
            Console.ForegroundColor = ConsoleColor.Red;
            Console.WriteLine("JULY and AUGUST are the central months of SUMMER");
            break;
        case "AU":
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.WriteLine("OCTOBER and NOVEMBER are the central months of AUTUMN");
            break;
        case "WI":
            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("JANUARY and FEBRUARY are the central months of WINTER");
            break;
        default:
            Console.WriteLine($"{seasonCode} is not a correct season code");
            break;
    }

    ...

}
```

Exercise “Genres”

Write the switch structure of a program that gives examples of videogames genres and subgenres.

GENRE	Sub-GENRE	EXAMPLE
Strategy (S)	Tactics (T)	Battle for Wesnoth
	Simulation (S)	Frostpunk
	Multiplayer Arenas (M)	Dota
Fighting (F)	Platformer (P)	Brawlhalla
	General (G)	Tekken
Life Simulation (L)	Social (S)	Castaway Paradise
	Digital Pets (D)	Pou
Puzzle (P)	Tile Matching (T)	Tetris
	Multiple Character Control (M)	Lemmings
	Hidden Object (H)	MineSweeper

Try different possibilities: A switch where some cases are also switches, a switch where some cases are if-else structures…

Screen captures on next page illustrate expected behaviour

```
char genre, subgenre;

Console.WriteLine("GENRE / SUBGENRE EXAMPLES");
Console.WriteLine("-------------------------");
Console.WriteLine();

Console.WriteLine("Enter genre...");
Console.Write("...Strategy(S) or Fighthing(F) or Life Simulation(L) or Puzzle(P) ");
genre = Convert.ToChar(Console.ReadLine());
Console.WriteLine();

switch (genre)
{
    case 'S':
        Console.WriteLine("and subgenre...");
        Console.Write("...Tacticts(T) or Simulation(S) or Multiplayer Arena(M) ");
        subgenre = Convert.ToChar(Console.ReadLine());
        switch(subgenre)
        {
            /* COMPLETE */
        }
        break;

     /* COMPLETE */
}
```

```
GENRE / SUBGENRE EXAMPLES
-------------------------

Enter genre...
...Strategy(S) or Fighthing(F) or Life Simulation(L) or Puzzle(P) J

Unknown genre J
```

```
GENRE / SUBGENRE EXAMPLES
-------------------------

Enter genre...
...Strategy(S) or Fighthing(F) or Life Simulation(L) or Puzzle(P) F

and subgenre...
...Platformer(P) or General(G) P

An example of Fighting/Platformer is WRAWLHALA
```

```
GENRE / SUBGENRE EXAMPLES
-------------------------

Enter genre...
...Strategy(S) or Fighthing(F) or Life Simulation(L) or Puzzle(P) L

and subgenre...
...Social(S) or Digital Pets(D) S

An example of Life Simulation/Social is CASTAWAY PARADISE
```

```
GENRE / SUBGENRE EXAMPLES
-------------------------

Enter genre...
...Strategy(S) or Fighthing(F) or Life Simulation(L) or Puzzle(P) S

and subgenre...
...Tacticts(T) or Simulation(S) or Multiplayer Arena(M) M

An example of Strategy/Multiplayer Arenas is DOTA
```

```
GENRE / SUBGENRE EXAMPLES
-------------------------

Enter genre...
...Strategy(S) or Fighthing(F) or Life Simulation(L) or Puzzle(P) F

and subgenre...
...Platformer(P) or General(G) U

Unknown subgenre U for Fighting
```

```
GENRE / SUBGENRE EXAMPLES
-------------------------

Enter genre...
...Strategy(S) or Fighthing(F) or Life Simulation(L) or Puzzle(P) P

and subgenre...
...Tile Matching(T) or Multiple Character Control(M) or Hidden Object(H) M

An example of Puzzle/Multiple Character Control is LEMMINGS
```

```
switch (genre)
{
    case 'S':
        Console.WriteLine("and subgenre...");
        Console.Write("...Tacticts(T) or Simulation(S) or Multiplayer Arena(M) ");
        subgenre = Convert.ToChar(Console.ReadLine());
        Console.WriteLine();
        switch(subgenre)
        {
            case 'T':
                Console.WriteLine("An example of Strategy/Tactics is BATTLE FOR WESNOTH"); break;
            case 'S':
                Console.WriteLine("An example of Strategy/Simulation is FROSTPUNK"); break;
            case 'M':
                Console.WriteLine("An example of Strategy/Multiplayer Arenas is DOTA"); break;
            default:
                Console.ForegroundColor = ConsoleColor.Red;
                Console.WriteLine("Unknown subgenre {0} for Strategy", subgenre);
                break;
        }
        break;


    case 'F':
        Console.WriteLine("and subgenre...");
        Console.Write("...Platformer(P) or General(G) ");
        subgenre = Convert.ToChar(Console.ReadLine());
        Console.WriteLine();
        switch (subgenre)
        {
            case 'P':
                Console.WriteLine("An example of Fighting/Platformer is WRAWLHALA"); break;
            case 'G':
                Console.WriteLine("An example of Fighting/General is TEKKEN"); break;
            default:
                Console.ForegroundColor = ConsoleColor.Red;
                Console.WriteLine("Unknown subgenre {0} for Fighting", subgenre);
                break;
        }
        break;

    case 'L':
        Console.WriteLine("and subgenre...");
        Console.Write("...Social(S) or Digital Pets(D) ");
        subgenre = Convert.ToChar(Console.ReadLine());
        Console.WriteLine();
        switch (subgenre)
        {
            case 'S':
                Console.WriteLine("An example of Life Simulation/Social is CASTAWAY PARADISE"); break;
            case 'D':
                Console.WriteLine("An example of Life Simulation/Digital Pets is POU"); break;
            default:
                Console.ForegroundColor = ConsoleColor.Red;
                Console.WriteLine("Unknown subgenre {0} for Life Simulation", subgenre);
                break;
        }
        break;

    case 'P':
        Console.WriteLine("and subgenre...");
        Console.Write("...Tile Matching(T) or Multiple Character Control(M) or Hidden Object(H) ");
        subgenre = Convert.ToChar(Console.ReadLine());
        Console.WriteLine();
        switch (subgenre)
        {
            case 'T':
                Console.WriteLine("An example of Puzzle/Tile Matching is TETRIS"); break;
            case 'M':
                Console.WriteLine("An example of Puzzle/Multiple Character Control is LEMMINGS"); break;
            case 'H':
                Console.WriteLine("An example of Puzzle/Hidden Object is MINESWEEPER"); break;
            default:
                Console.ForegroundColor = ConsoleColor.Red;
                Console.WriteLine("Unknown subgenre {0} for Puzzle", subgenre);
                break;
        }
        break;

    default:
        Console.ForegroundColor = ConsoleColor.Red;
        Console.WriteLine("Unknown genre {0} ", genre);
        break;
}
```

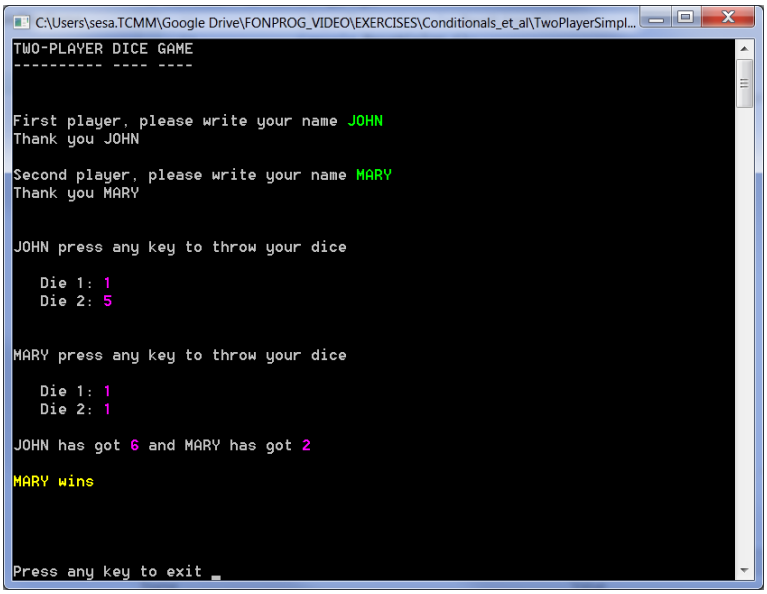
Exercise “Simple dice game”

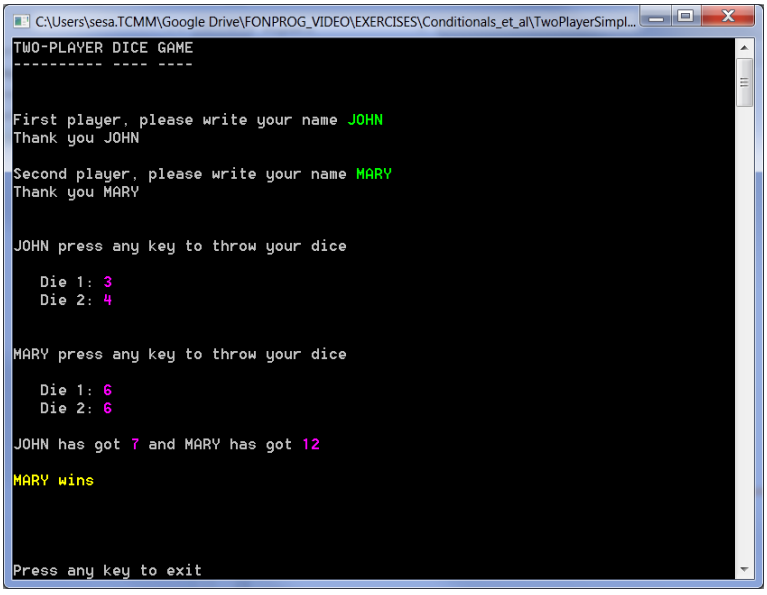
Write (complete) a C# program for playing a two-player simple dice game. The rules are very simple:

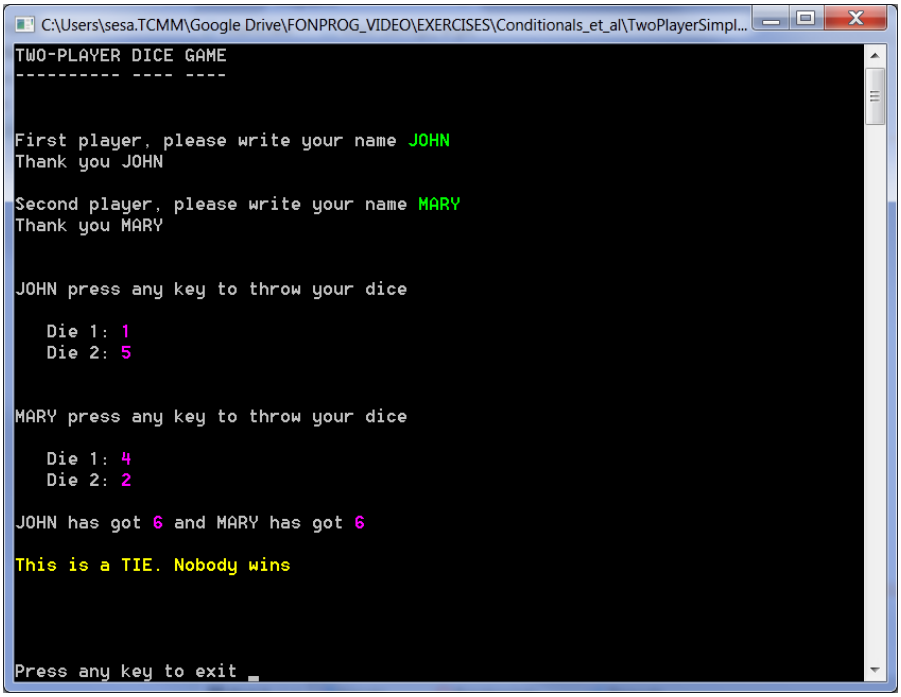
- Each player has two dice
- First player throws their dice. Then second player follows suit.
- The winner is the player who gets a higher number. If they get the same number then that’s a tie and nobody wins.

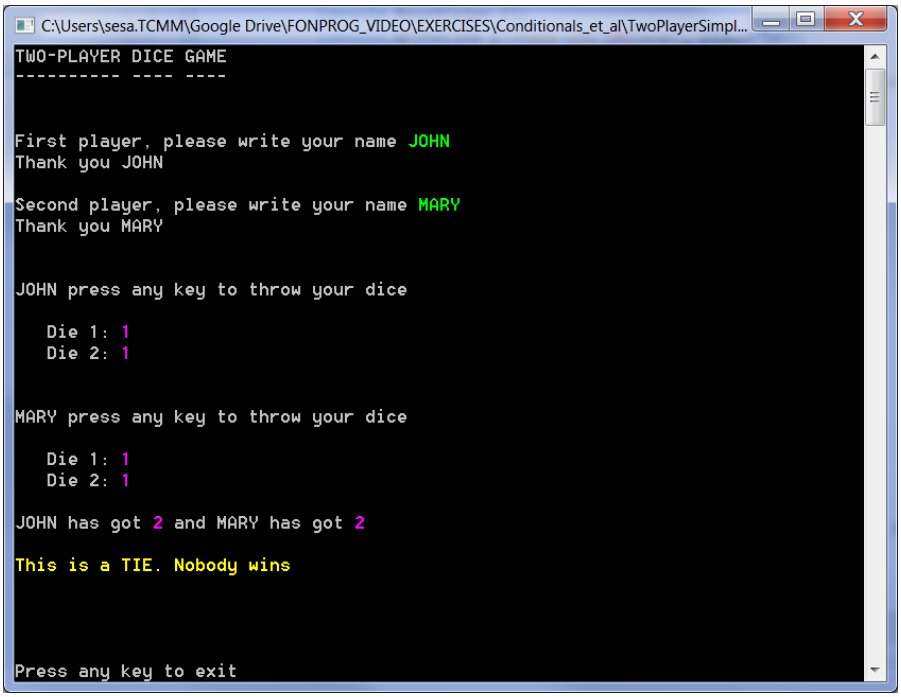
1+1 (=2) is called SNAKE EYES and is considered to be higher than any other number.

The “trickiest” part of this problem is deciding the winner. Notice that 1+1(=2) is higher than, say, 4+5(=9). If just before making the comparison that determines the winner, 2 is converted into 13 (or any other number greater than 12) then the problem becomes easier. **BUT, you are requested to solve the problem WITHOUT doing such a conversion.**









Next pages show different approaches that can be useful when solving the problem.

APPROACH 1

```
if ( PLAYER 1 GETS SNAKE EYES) {
    // THERE TWO CASES TO CONSIDER
}
else {
    // ALL OTHER CASES (PLAYER TWO WINS
    - WITH OR WITHOUT SNAKE EYES -, PLAYER
      ONE WINS, THERE'S A TIE
}
```

INSIDE THE ELSE BRANCH DEVELOP A NESTED IF-ELSE STRUCTURE TO COVER THE DIFFERENT POSSIBILITIES

APPROACH 2

```
if ( PLAYER 1 WINS ) {
    // PLAYER 1 WINS
}
else {
    // ONLY TWO POSSIBILITIES REMAIN:
      - THERE IS A TIE
      - PLAYER 2 WINS
}
```

IN THIS APPROACH THE "COMPLEXITY" OF THE PROBLEM HAS BEEN MOVED TO THE CONDITION. HOW CAN WE EXPRESS THAT PLAYER 1 WINS (WITH OR WITHOUT SNAKE EYES)?

APPROACH 3

CONSIDER THE FOLLOWING DISJOINT POSSIBILITIES

- PLAYER ONE GETS SNAKE EYES AND PLAYER TWO DOES NOT
- PLAYER TWO GETS " " AND PLAYER ONE " "
- NONE OF THE ABOVE HOLD AND PLAYER ONE HAS MORE POINTS
- " " " " " " PLAYER TWO " " "
- NONE OF THE ABOVE (TIE)

APPROACH 4

CONSIDER THE FOLLOWING DISJOINT POSSIBILITIES

- PLAYERS HAVE THE SAME POINTS (TIE)
- PLAYERS DON'T HAVE THE SAME POINTS AND PLAYER ONE HAS SNAKE EYES (PLAYER ONE WINS)
- PLAYERS DON'T HAVE THE SAME POINTS AND PLAYER TWO HAS SNAKE EYES (PLAYER TWO WINS)
- THERE'S NO TIE, NO PLAYER HAS SNAKE EYES AND PLAYER ONE HAS MORE POINTS (PLAYER ONE WINS)
- NONE OF THE ABOVE HOLDS (PLAYER TWO WINS)

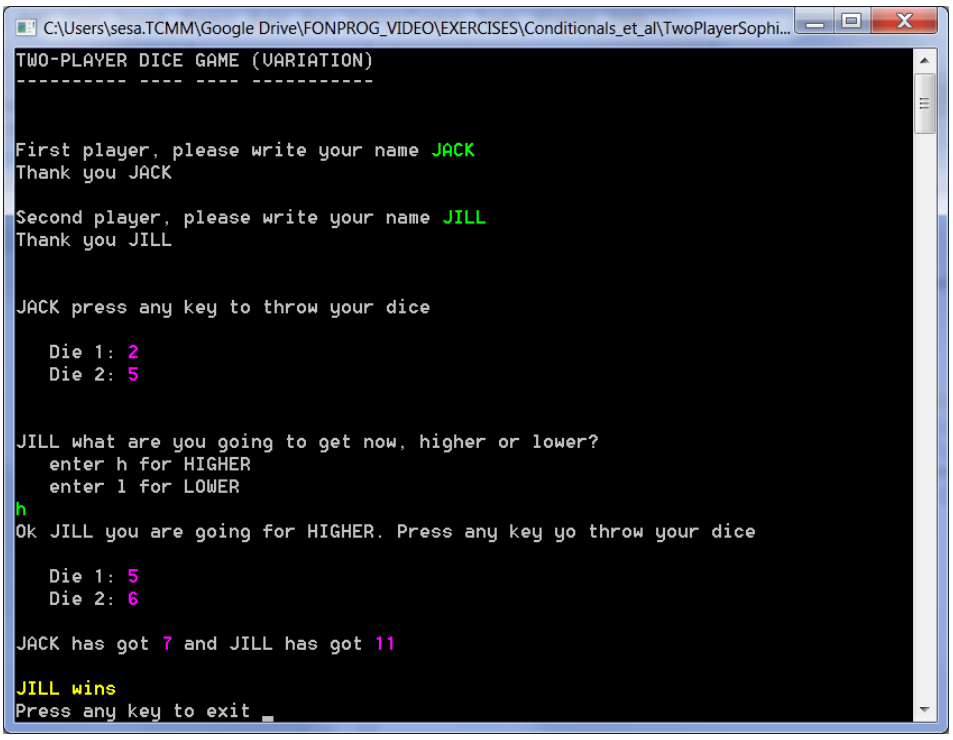
```
// Simple dice game. Solution based on approach #4
public static void Main (string[] args)
{
    String playerOne, playerTwo;
    int dieOnePlayerOne, dieTwoPlayerOne;
    int dieOnePlayerTwo, dieTwoPlayerTwo;
    int totalOne, totalTwo;
    Random randomNumberGenerator;

    // create and store the random number generator
    randomNumberGenerator = new Random ();

    ...

    dieOnePlayerOne = randomNumberGenerator.Next (1, 7);
    dieTwoPlayerOne = randomNumberGenerator.Next (1, 7);

    ...

    dieOnePlayerTwo= randomNumberGenerator.Next (1, 7);
    dieTwoPlayerTwo = randomNumberGenerator.Next (1, 7);

    ...

    totalOne = dieOnePlayerOne + dieTwoPlayerOne;
    totalTwo = dieOnePlayerTwo + dieTwoPlayerTwo;

    ...

    Console.WriteLine ('\n');
    Console.ForegroundColor = ConsoleColor.Yellow;
    if (totalOne == totalTwo) {
        Console.WriteLine ("This is a TIE. Nobody wins");
    } else if (totalOne == 2) {
        // player one has got snake eyes
        Console.WriteLine ($"{playerOne} wins");
    } else if (totalTwo == 2) {
        // player two has got snake eyes
        Console.WriteLine ($"{playerTwo} wins");
    } else if (totalOne > totalTwo) {
        Console.WriteLine ($"{playerOne} wins");
    } else {
        Console.WriteLine ($"{playerTwo} wins");
    }
    ...
}
```

Exercise "Less simple dice game"

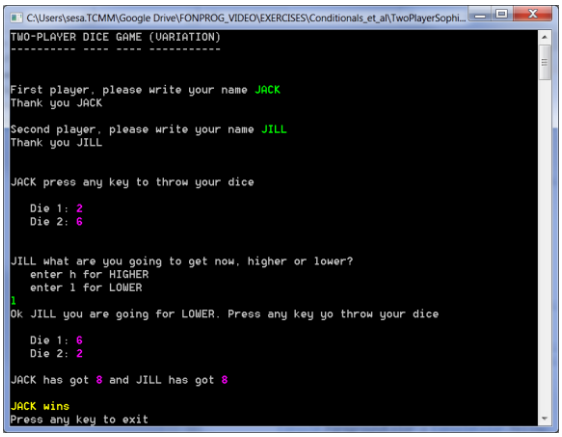
Write (complete) a C# program for playing a variation of the simple dice game. These are the new rules:

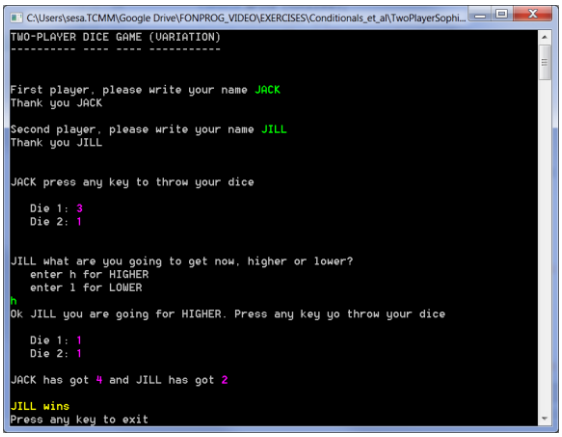
- Each player has two dice
- First player throws his dice.
- Second player bets on HIGHER (he thinks he's going to get a higher number) or on LOWER (he thinks he's going to get a lower number).
- After having put his bet, second player throws his dice.
- If second player had bet on HIGHER and he has got a higher number, he wins. Otherwise first player wins. Similar for LOWER
- 1+1 (=2) is called SNAKE EYES and is considered to be higher than any other number.

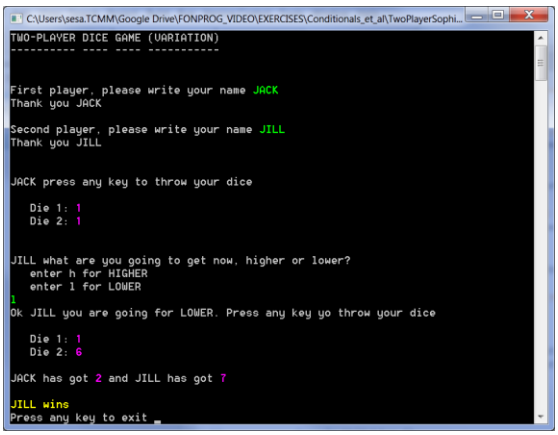
You can solve the problem TWICE: first changing 2 into 13 and later without this conversion



More screen captures on next page


C:\Users\sesa.TCMM\Google Drive\FONPROG_VIDEO\EXERCISES\Conditionals_et_al\TwoPlayerSophi...
TWO-PLAYER DICE GAME (VARIATION)
First player, please write your name JACK
Thank you JACK
Second player, please write your name JILL
Thank you JILL
JACK press any key to throw your dice
Die 1: 2
Die 2: 6
JILL what are you going to get now, higher or lower?
enter h for HIGHER
enter l for LOWER
l
Ok JILL you are going for LOWER. Press any key yo throw your dice
Die 1: 6
Die 2: 2
JACK has got 8 and JILL has got 8
JACK wins
Press any key to exit


C:\Users\sesa.TCMM\Google Drive\FONPROG_VIDEO\EXERCISES\Conditionals_et_al\TwoPlayerSophi...
TWO-PLAYER DICE GAME (VARIATION)
First player, please write your name JACK
Thank you JACK
Second player, please write your name JILL
Thank you JILL
JACK press any key to throw your dice
Die 1: 3
Die 2: 1
JILL what are you going to get now, higher or lower?
enter h for HIGHER
enter l for LOWER
h
Ok JILL you are going for HIGHER. Press any key yo throw your dice
Die 1: 1
Die 2: 1
JACK has got 4 and JILL has got 2
JILL wins
Press any key to exit


C:\Users\sesa.TCMM\Google Drive\FONPROG_VIDEO\EXERCISES\Conditionals_et_al\TwoPlayerSophi...
TWO-PLAYER DICE GAME (VARIATION)
First player, please write your name JACK
Thank you JACK
Second player, please write your name JILL
Thank you JILL
JACK press any key to throw your dice
Die 1: 1
Die 2: 1
JILL what are you going to get now, higher or lower?
enter h for HIGHER
enter l for LOWER
l
Ok JILL you are going for LOWER. Press any key yo throw your dice
Die 1: 1
Die 2: 6
JACK has got 2 and JILL has got 7
JILL wins
Press any key to exit

```
// Less simple dice game. Solution
public static void Main (string[] args)
{
    ...

    /* COMPLETE FROM THIS POINT */

    Console.WriteLine ();
    Console.Write ("   Die 1: ");
    Console.ForegroundColor = ConsoleColor.Magenta;
    Console.WriteLine (dieOnePlayerTwo);
    Console.ResetColor ();
    Console.Write ("   Die 2: ");
    Console.ForegroundColor = ConsoleColor.Magenta;
    Console.WriteLine (dieTwoPlayerTwo);
    Console.ResetColor ();
    totalOne = dieOnePlayerOne + dieTwoPlayerOne;
    totalTwo = dieOnePlayerTwo + dieTwoPlayerTwo;
    Console.WriteLine ();
    Console.Write ($"{playerOne} has got ");
    Console.ForegroundColor = ConsoleColor.Magenta;
    Console.Write (totalOne);
    Console.ResetColor ();
    Console.Write ($" and {playerTwo} has got ");
    Console.ForegroundColor = ConsoleColor.Magenta;
    Console.Write (totalTwo);
    Console.ResetColor ();

    Console.WriteLine ('\n');
    Console.ForegroundColor = ConsoleColor.Yellow;
    switch (election) {
    case 'l':
        if (totalTwo < totalOne || (totalTwo!=2 && totalOne==2)) {
            Console.WriteLine ($"{playerTwo} wins");
        } else {
            Console.WriteLine ($"{playerOne} wins");
        }
        break;
    case 'h':
        if ((totalTwo > totalOne && totalOne!=2) || (totalOne!=2 && totalTwo==2)) {
            Console.WriteLine ($"{playerTwo} wins");
        } else {
            Console.WriteLine ($"{playerOne} wins");
        }
        break;
    default:
        Console.WriteLine ($"{playerTwo} loses because he/she has pressed an incorrect key");
        break;
    }
    ...
}
```

Exercise "Course mark calculator"

In a university course students get six different marks:

- midterm exam (mte)
- end of term exam (ete)
- practical assignment #1 (pa1)
- practical assignment #2 (pa2)
- practical assignment #3 (pa3)
- practical assignment #4 (pa4)

From these marks the course final mark (CFM) is computed as follows

$$ex = \begin{cases} max(\,(mte + ete)/2, ete) & if ete \geq 4 \\ ete & otherwise \end{cases}$$

$$pa = \begin{cases} (pa1 + pa2 + pa3 + pa4)/4 & if\ all\ pa_i\ are \geq 4 \\ 0 & if any\ pa_i\ is < 2.5 \\ pa4 & otherwise \end{cases}$$

$$CFM = \begin{cases} 0.75 \cdot ex + 0.25 \cdot pa & if\ both\ pa\ and\ ex\ are \geq 5 \\ 0.65 \cdot ex + 0.35 \cdot pa & if ex \geq 5\ \ but\ \ pa < 5 \\ ex & otherwise \end{cases}$$

Write (complete) a C# program for computing ex, pa and CFM from mte, ete, and PA_i

Notice that the rules to calculate ex, pa and CFM can be easily translated into if/else-like structures.

max((mte + ete)/2, ete) means the maximum between *(mte+ete)/2* and *ete*

```
COURSE MARK CALCULATOR
------ ---- ----------

Please enter 'mte' 5
Please enter 'ete' 4
Please enter 'pa1' 4
Please enter 'pa2' 5
Please enter 'pa3' 6
Please enter 'pa4' 7

==> ex = 4.5

==> pa = 5.5

==> CFM = 4.5
```

```
COURSE MARK CALCULATOR
------ ---- ----------

Please enter 'mte' 4
Please enter 'ete' 8
Please enter 'pa1' 2.4
Please enter 'pa2' 8
Please enter 'pa3' 8
Please enter 'pa4' 8

==> ex = 8

==> pa = 0

==> CFM = 5.2
```

```
COURSE MARK CALCULATOR
------ ---- ----------

Please enter 'mte' 2
Please enter 'ete' 8
Please enter 'pa1' 5
Please enter 'pa2' 6
Please enter 'pa3' 7
Please enter 'pa4' 3

==> ex = 8

==> pa = 3

==> CFM = 6.25
```

```
COURSE MARK CALCULATOR
------ ---- ----------

Please enter 'mte' 6
Please enter 'ete' 7
Please enter 'pa1' 6
Please enter 'pa2' 6
Please enter 'pa3' 4
Please enter 'pa4' 4

==> ex = 7

==> pa = 5

==> CFM = 6.5
```

```
COURSE MARK CALCULATOR
------ ---- ----------

Please enter 'mte' 4
Please enter 'ete' 2
Please enter 'pa1' 5
Please enter 'pa2' 6
Please enter 'pa3' 7
Please enter 'pa4' 8

==> ex = 2

==> pa = 6.5

==> CFM = 2
```

```
COURSE MARK CALCULATOR
------ ---- ----------

Please enter 'mte' 4
Please enter 'ete' 3.5
Please enter 'pa1' 6
Please enter 'pa2' 1
Please enter 'pa3' 8
Please enter 'pa4' 9

==> ex = 3.5

==> pa = 0

==> CFM = 3.5
```

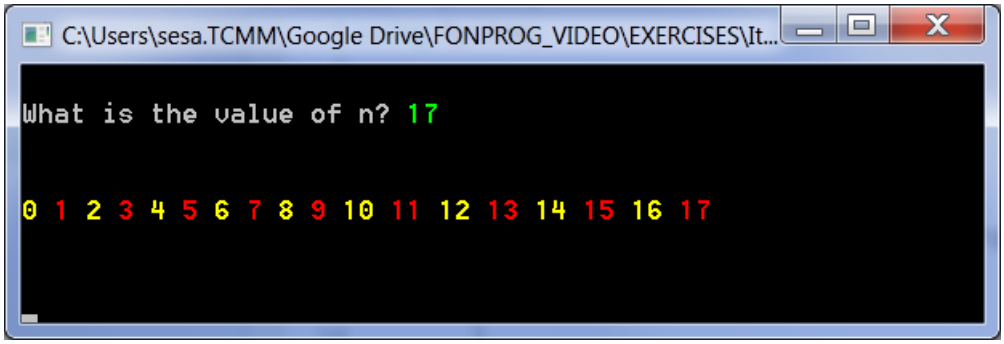
```
COURSE MARK CALCULATOR
------ ---- ----------

Please enter 'mte' 4
Please enter 'ete' 4
Please enter 'pa1' 3
Please enter 'pa2' 4
Please enter 'pa3' 5
Please enter 'pa4' 6

==> ex = 4

==> pa = 6

==> CFM = 4
```

```
// Course mark calculator. Solution
public static void Main (string[] args)
{
    double mte, ete;
    double pa1, pa2, pa3, pa4;
    double ex, pa;
    double CFM;
    ...
    // compute ex
    if (ete>=4) { ex = Math.Max((mte+ete)/2, ete);}
    else { ex = ete;}

    // compute pa
    if (pa1>=4 && pa2>=4 && pa3>=4 && pa4>=4) { pa = (pa1 + pa2 + pa3 + pa4) / 4;}
    else if (pa1<2.5 || pa2<2.5 || pa3<2.5 || pa4<2.5) { pa = 0;}
    else { pa = pa4; }

    // compute CFM
    if (pa>=5 && ex>=5) { CFM = 0.75 * ex + 0.25 * pa; }
    else if (ex>=5 && pa<5) { CFM = 0.65 * ex + 0.35 * pa; }
    else { CFM = ex;}

    // Show results
    Console.ForegroundColor = ConsoleColor.White;
    Console.BackgroundColor = ConsoleColor.Blue;
    Console.WriteLine();
    Console.WriteLine();
    Console.WriteLine(" ==> ex = " + ex);
    Console.WriteLine();
    Console.WriteLine(" ==> pa = " + pa);
    Console.WriteLine();
    Console.WriteLine(" ==> CFM = " + CFM);
    ...
}
```

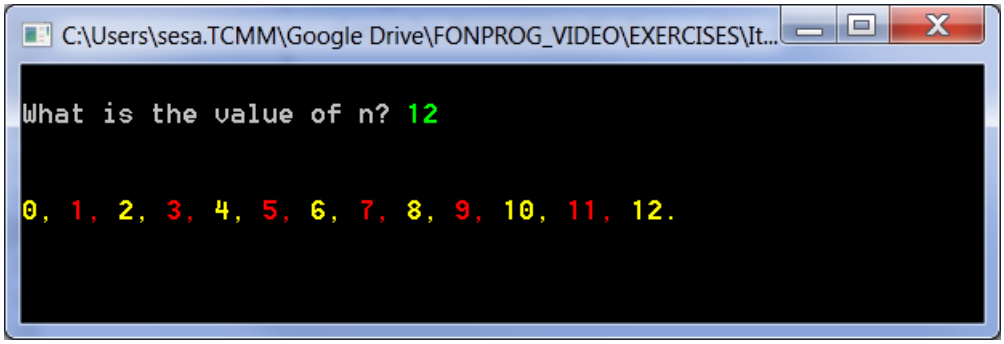
ITERATIVE EXECUTION

In the following exercises use for-based iterations

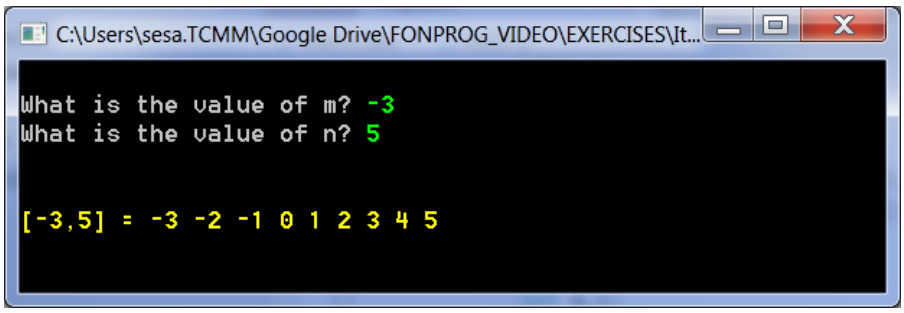
1. Write a program that given an int number n (n>0) writes all numbers from 0 until n (inclusive). Even numbers are written in yellow while odd numbers are written in red. DO NOT verify that n>0 (trust the user). The following image shows the result produced when the number given is 17.

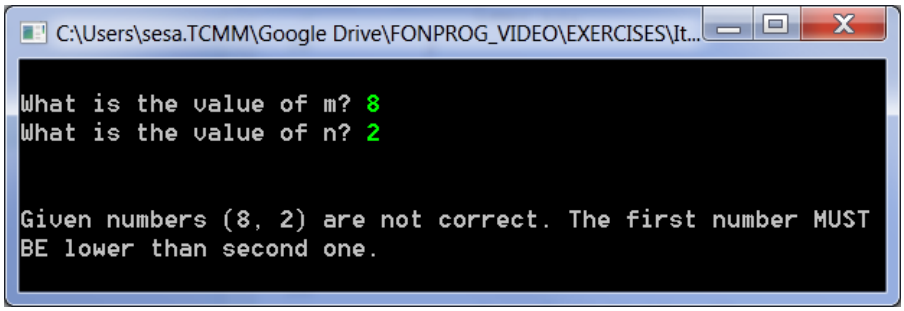


2. Same as before, but now the numbers are separated by commas. There's a period after the last number. The following image shows the result produced when the number given is 12.

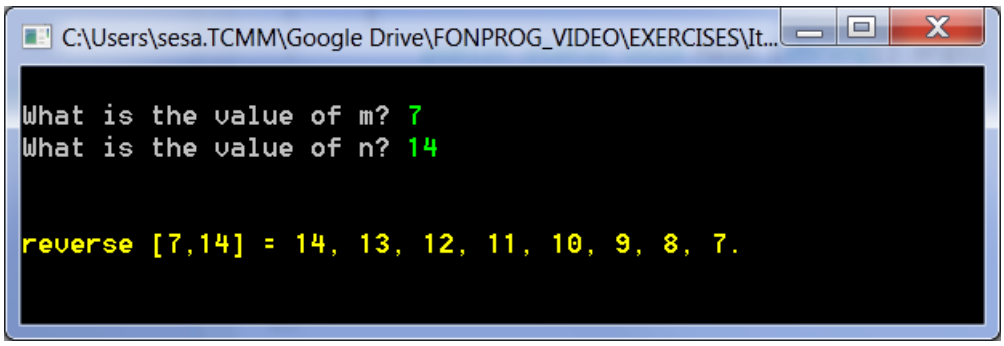


3. Write a program that given two int numbers m and n (with m < n) writes, in ascending order, all the numbers in the interval [m, n]. The program MUST verify that the first number (m) i lower than the second (n). IF it is not, the program must warn its user and terminate. The following images illustrate the behaviour of the program.

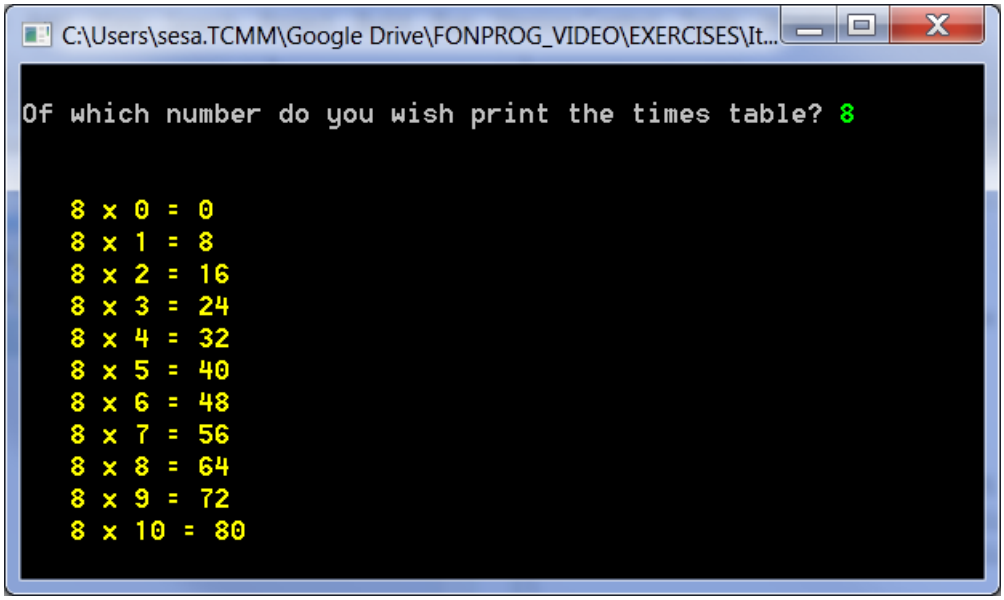




4. Same as before but now in decreasing order and separating the numbers with commas. There must be a period after the last number. The following image shows the result produced when the numbers given are 7 and 14.



5. Write a program that given an int number prints the times table of that number. For instance if the given number is 8, it writes the 8 times table.



Solutions of exercises 1 to 5 start on next page

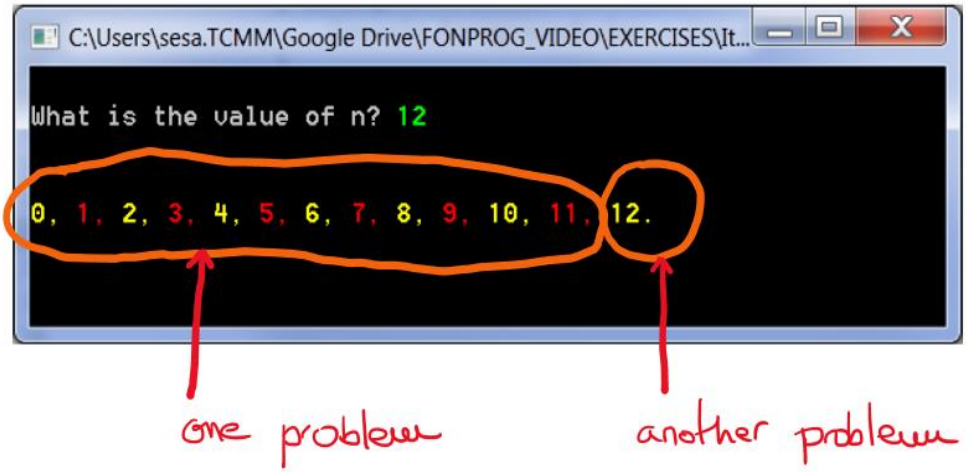
Exercise 1

```
for (int i=0; i<=n; i=i+1)
    // Iterate from 0 up to n (included)
{
    // Select colour depending on parity
    if (i % 2 == 0) {
        Console.ForegroundColor = ConsoleColor.Yellow;
    } else {
        Console.ForegroundColor = ConsoleColor.Red;
    }

    // write the number (with a space)
    Console.Write (i + " "); // same as Console.Write ($"{i} ");
}
```

Exercise 2

```
for (int i = 0; i <= n; i = i + 1)
// iterate from 0 up to n (included)
{
    // select color depending on parity
    if (i % 2 == 0) { Console.ForegroundColor = ConsoleColor.Yellow; }
    else { Console.ForegroundColor = ConsoleColor.Red; }

    // write the number
    Console.Write(i);
    // write the number's "companion". All but the last (n) have ,
    if (i!=n) { Console.Write(", "); }
    else { Console.Write(". "); }
}
```

There are other possibilities. For instance: break down the problem into two smaller parts (subproblems):




```
for (int i = 0; i < n; i = i + 1)
// iterate from 0 up to n-1 (notice i<n excludes n)
{
    // select color depending on parity
    if (i % 2 == 0) { Console.ForegroundColor = ConsoleColor.Yellow; }
    else { Console.ForegroundColor = ConsoleColor.Red; }

    // write the number and its "companion" (always a , followed by a space)
    Console.Write(i + ", "); // equivalent to Console.Write("{0}, ",i);

}

// finish the sequence with n.
if (n % 2 == 0) { Console.ForegroundColor = ConsoleColor.Yellow; }
else { Console.ForegroundColor = ConsoleColor.Red; }
Console.Write(n + ". ");
```

Exercise 3

```
if (m>=n)
{
    // complain
    Console.Write("Given numbers ({0}, {1}) are not correct. ",m,n);
    Console.WriteLine("The first nummber MUST BE lower than the second");
}
else
{
    // m and n are correct
    Console.ForegroundColor = ConsoleColor.Yellow;
    Console.Write("[{0}, {1}] = ", m, n);
    for (int i=m; i<=n; i=i+1)
        // iterate from m up to n (both included)
    {
        // write number and a white space
        Console.Write(i + " ");
    }
}
```

UNFOLD [m,n]

Exercise 4

```
if (m>=n)
{
    // complain
    Console.Write("Given numbers ({0}, {1} are not correct. ", m, n);
    Console.WriteLine("The first number MUST BE lower than the second");
}
else
{
    // m and n are correct
    Console.ForegroundColor = ConsoleColor.Yellow;
    Console.Write("reverse [{0}, {1}] = ", m, n);
    for (int i=n; i>=m; i=i-1)
        // iterate from n up to m IN DECREASING ORDER
    {
        // write the number...
        Console.Write(i);
        // ... and its companions
        if (i==m) { Console.Write("."); }
        else { Console.Write(", "); }
    }
}
```

Exercise 5

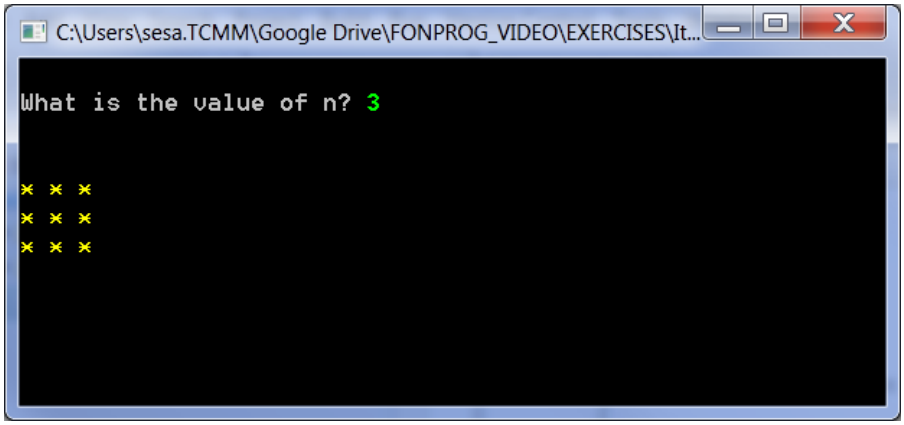
```
Console.ForegroundColor = ConsoleColor.Yellow;
for (int multiplier = 0; multiplier<=10; multiplier++)
    // unfold interval [0, 10]
{
    Console.WriteLine("{0} x {1} = {2}", n, multiplier, n * multiplier);
}
```

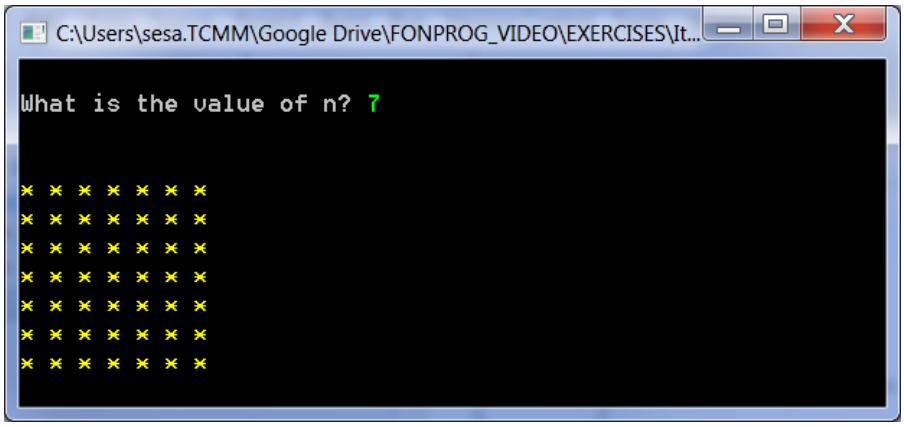
2D-shape “drawing” exercises

Exercises from 6 to 20 have a workout nature. Solve them using for-based iterations. Later, when you know about while-based iterations try to solve them again using this type of iterations. In all cases give priority to the ones recommended by your instructor. Solve one out of every two, at least, if you think you have not time to solve them all.

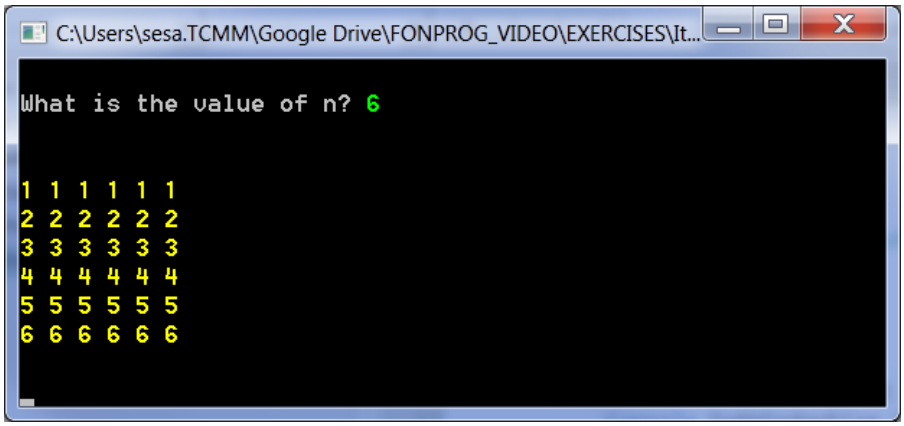
Solutions at the end of the section

6. Write a program that given an int number n (n>=1; trust the user) draws an nxn square of asterisks (*). The following images exemplify the behaviour of the program. Notice that there’s a blank space after each asterisk. DO NOT USE cursor positioning!

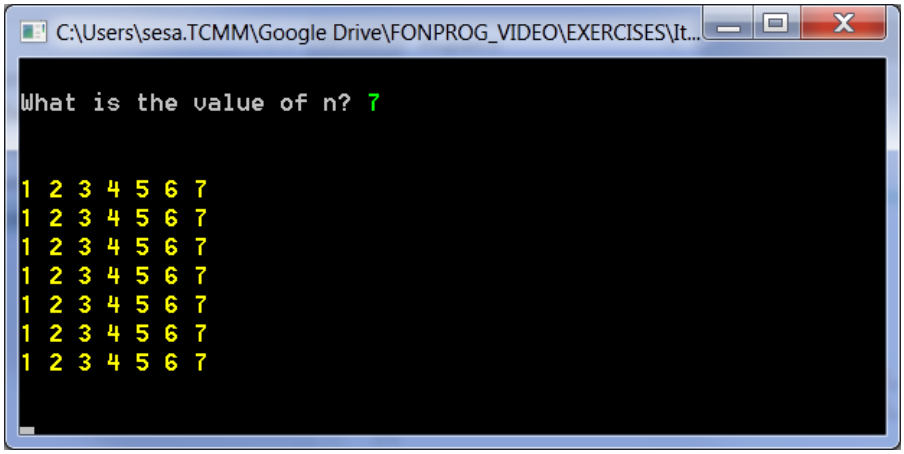




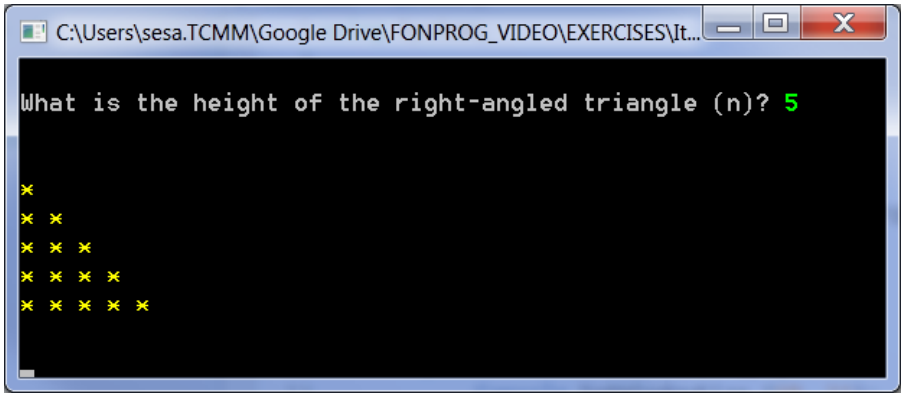
7. Write a program that given an int number n (1<=n<=9; trust the user) draws an nxn square made of the numbers from 1 to n (each row is made of n repetitions of the number of the row). The following image exemplifies the behaviour of the program. Notice that there's a blank space after each number. DO NOT USE cursor positioning!



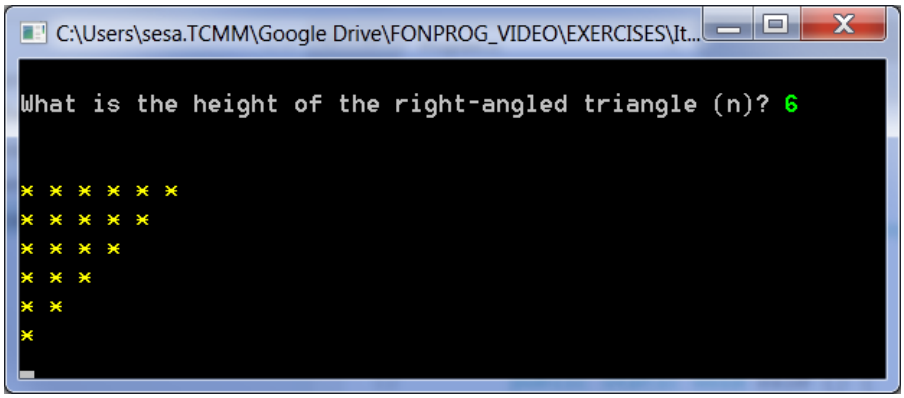
8. Same as before but now each row is made of the numbers from 1 to n



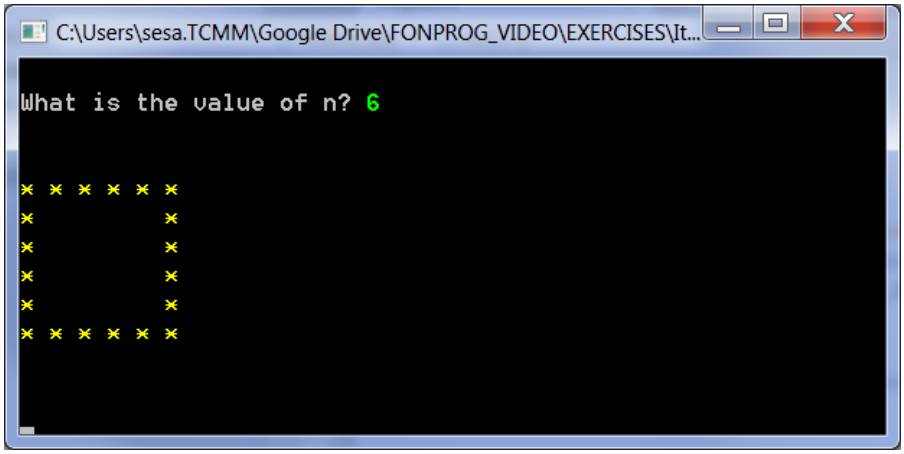
9. Write a program that given an int number n (n>=1) draws a right-angled triangle of height n made of asterisks. The program MUST check that the number is greater than 0. If it is not, the program must warn its user and terminate. Do not use cursor positioning. The following image shows a triangle of height 5.



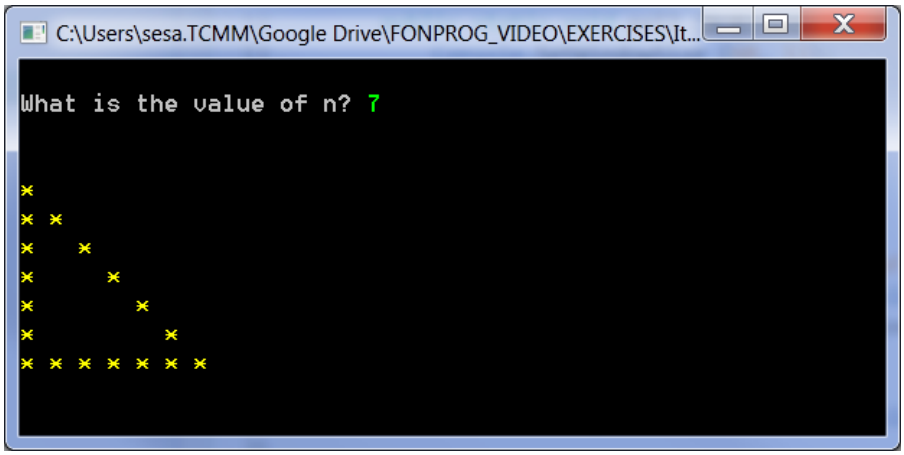
10. Same as before but with the base facing upwards.



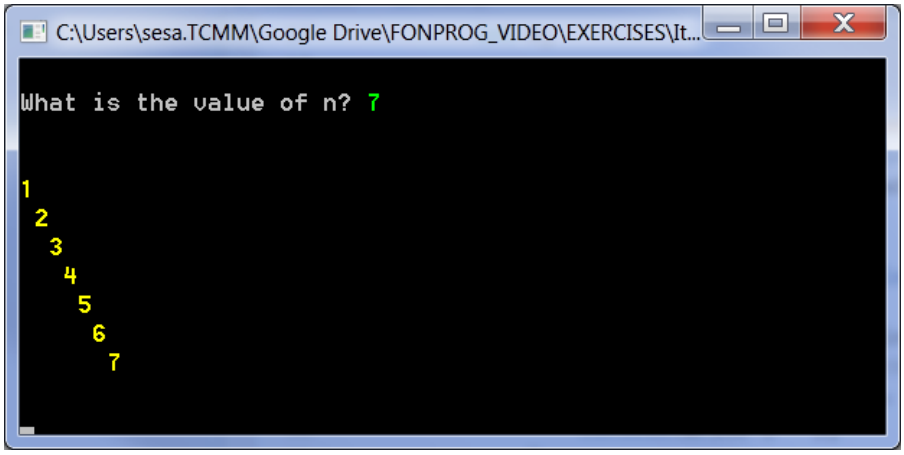
11. Write a program that given an int number n (n>=2, trust the user) draws the perimeter of an nxn square. Do not use cursor positioning. The following image shows the result for n=6.



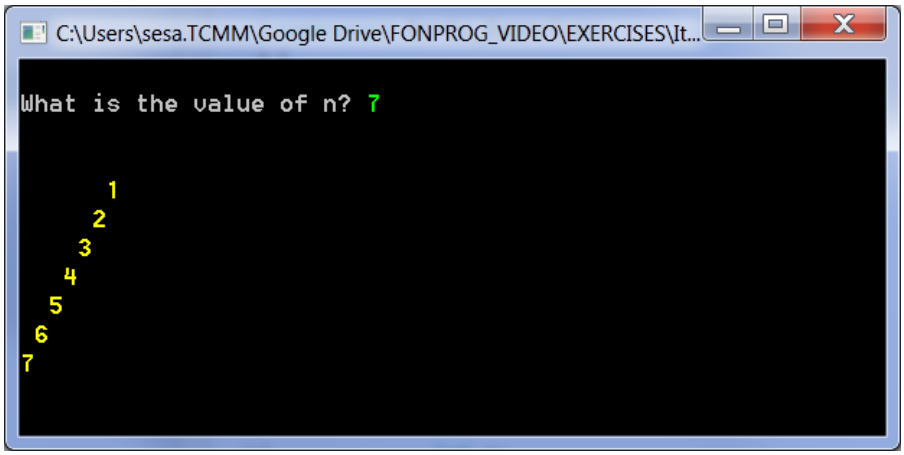
12. Write a program that given an int number n (n>=2, trust the user) draws the perimeter of a right-angled triangle of height n. Do not use cursor positioning. The following image shows the result for n=7.



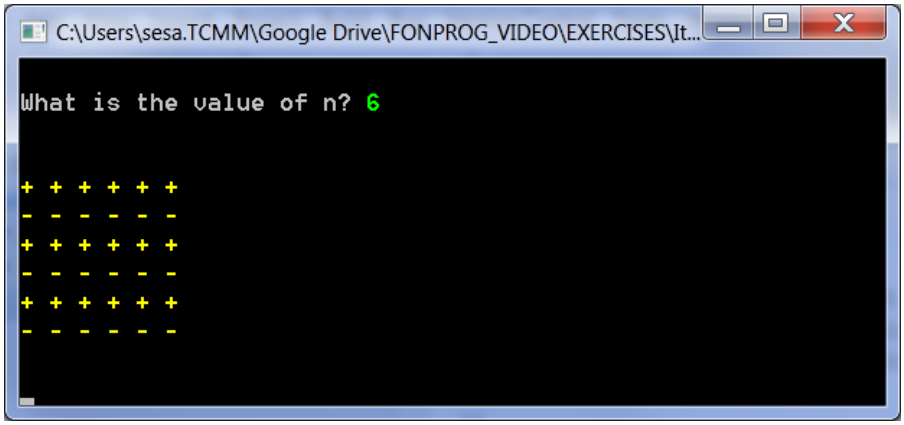
13. Write a program that given an int number n (n>=1, trust the user) draws a left-to-right diagonal made of the numbers from 1 to n. Do not use cursor positioning. The following image shows the result for n=7.



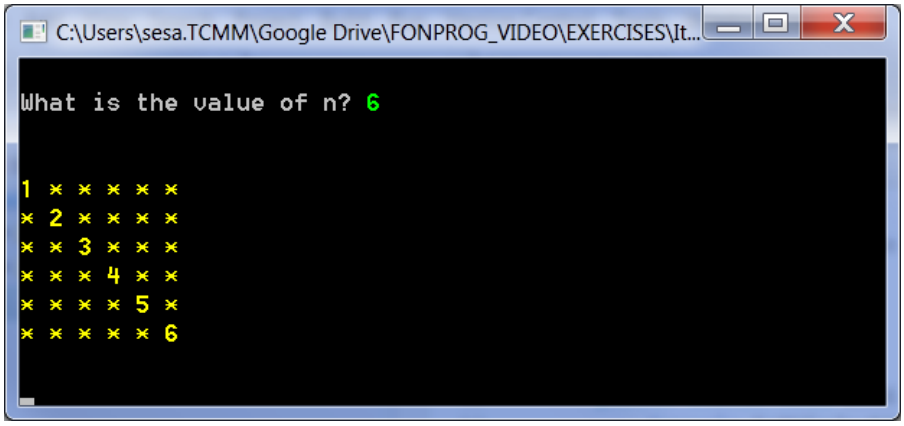
14. Same as before but now the diagonal must run right-to-left



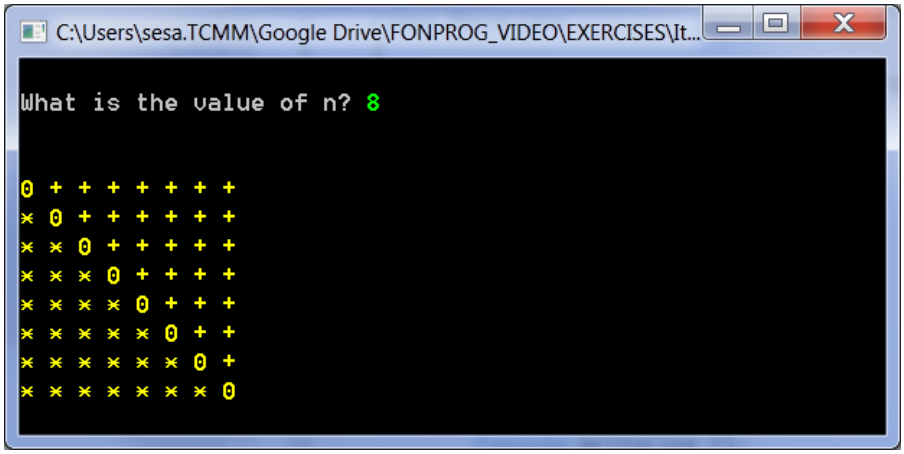
15. Write a program that given an int number n (n>=1; trust the user) draws an nxn square. Odd lines must be made of plus signs (+) while the even ones must be made of minus signs (-). DO NOT USE cursor positioning. The following image shows the result for n=6



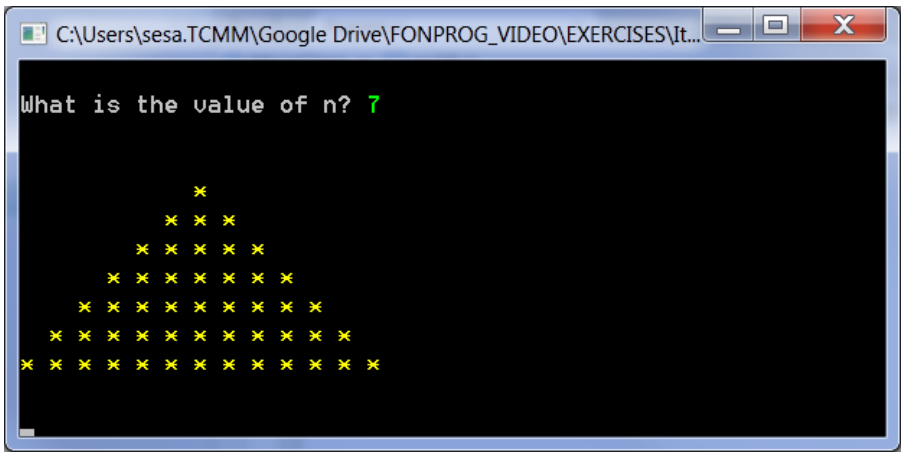
16. Write a program that given an int number n (n>=1; trust the user) draws an nxn square made of asterisk, except for the left-to-right diagonal that must contain the number of the line . DO NOT USE cursor positioning. The following image shows the result for n=6



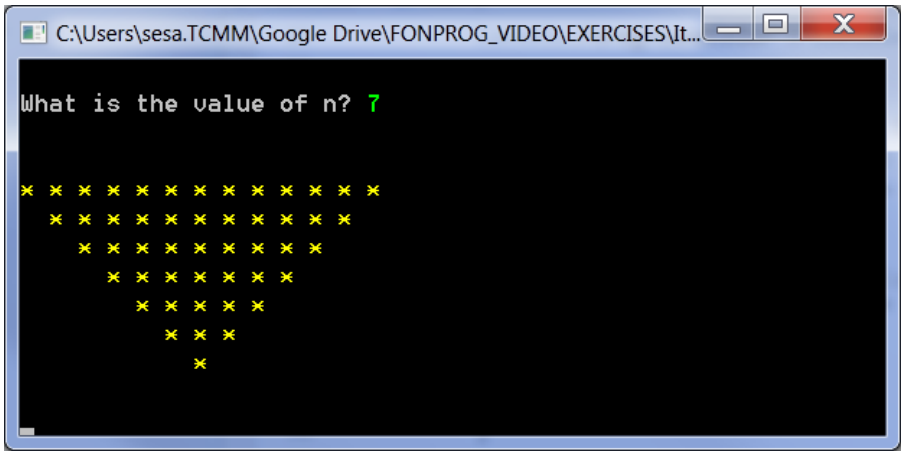
17. Write a program that given an int number n (n>=1; trust the user) draws an nxn square with the following characteristics: the left-to-right diagonal is made of zeros (0), the area above this diagonal is made of plus signs (+) and the area below is made of asterisks (*).DO NOT USE cursor positioning. The following image shows the result for n=8



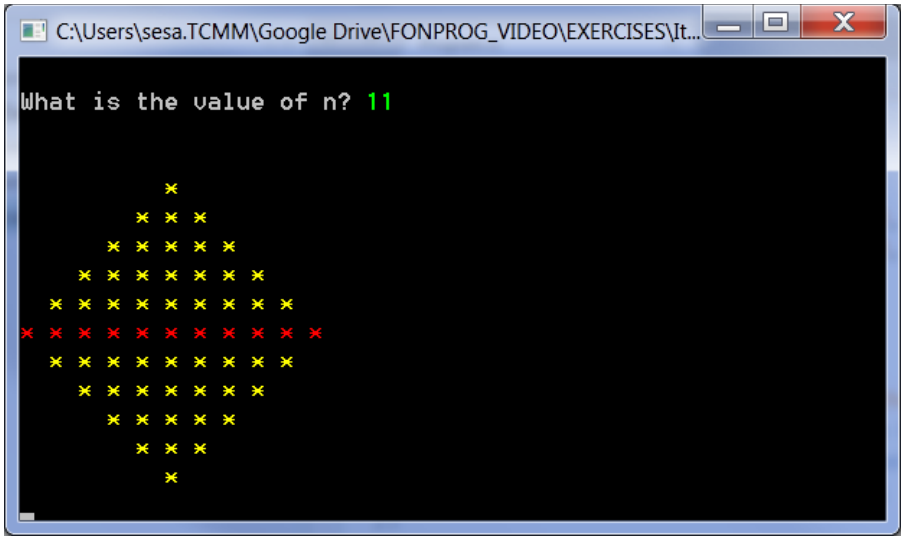
18. Write a program that given an int number n (n>=1, trust the user) draws an isosceles triangle of height n made of asterisks. Do not use cursor positioning. The image below shows a triangle of height 7.
    Hints:
    - Determine the number of asterisks in each line (as a function of the number of the line).
    - Determine the number of blanks that precede the sequence of asterisks (as function of the number of the line and n).



19. Same as before but now upside-down.
    Hint: there are 2·n – 2(line-1) -1 = 2·(n-line) 1 asterisks in each line. You should determine, for each line, how many blank spaces precede the first asterisk



20. Write a program that given an odd int number n (n>=1 n is odd, trust the user) draws a diamond height (and width) n made of asterisks. Use a different colour for the central line. Do not use cursor positioning. The image below shows a diamond of height 11.
    Hint: break the shape into three parts...



Exercise 7

```
for (int line = 1; line <=n; line++)
{

    // Write a line of numbers
    for (int counter = 1; counter <= n; counter++)
        // repeat n times write the number in line
    {
        Console.Write(line + " ");
    }
    Console.WriteLine();

}
```

Repeat n times
write the
number of
the line

```
1 1 1 1 1 1
2 2 2 2 2 2
3 3 3 3 3 3
4 4 4 4 4 4
5 5 5 5 5 5
6 6 6 6 6 6
```

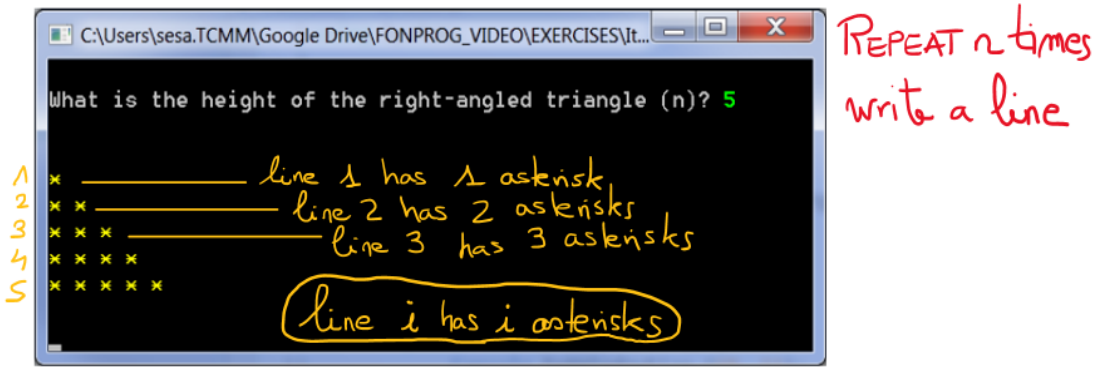
Exercise 8

```
for (int line = 1; line <=n; line++)
{

    // Write a line of numbers
    for (int num = 1; num <= n; num++)
        // write all numbers from 1 to n
    {
        Console.Write(num + " ");
    }
    Console.WriteLine();

}
```

Repeat n times
write numbers
from 1 to n

```
1 2 3 4 5 6 7
1 2 3 4 5 6 7
1 2 3 4 5 6 7
1 2 3 4 5 6 7
1 2 3 4 5 6 7
1 2 3 4 5 6 7
1 2 3 4 5 6 7
```

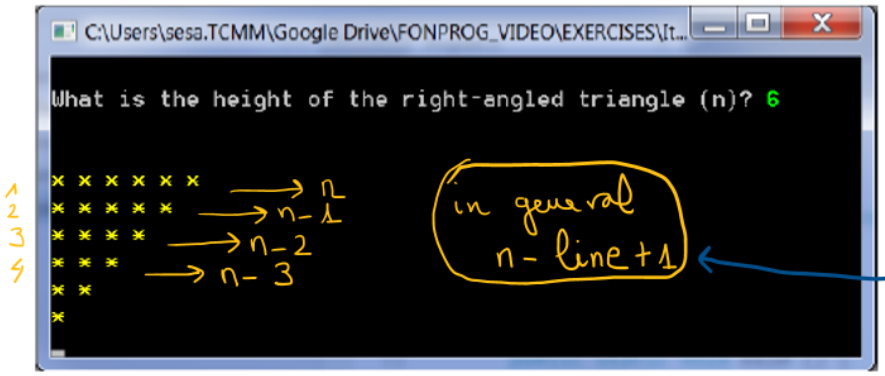
Exercise 9



Each line has as many asterisks as the number of the line

```
for (int line=1; line<=n; line++)
    // the triangle has n lines (from 1 to n)
{
    for (int ast=1; ast<=line; ast++)
        // each line has as many asteriks
        // as the number of the line
    {
        Console.Write("* ");
    }

    // jump and get ready for the next line
    Console.WriteLine();
}
```

Exercise 10

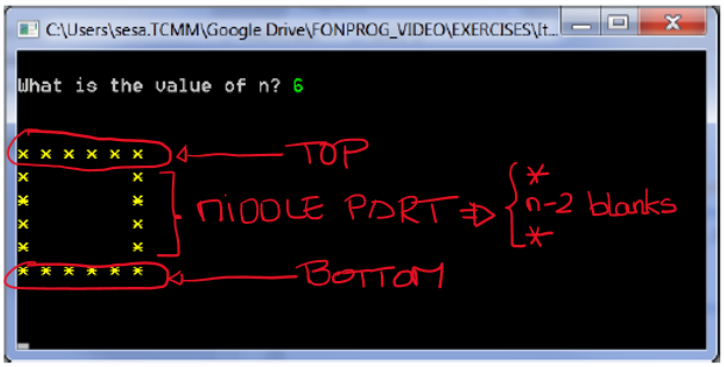


A triangle of height n has n lines.
repeat n times
write a line of (?) asterisks

the question is: how many asterisks are there in each line?

```
if (n <= 0) {
    Console.WriteLine ("Given number ({0}) is not correct.", n);
    Console.WriteLine ("A positive (>=1) number was expected");
} else {

    for (int line=1; line<=n; line++)
        // the triangle has n lines (from 1 to n)
    {
        for (int ast=1; ast<=n-line+1; ast++)
            // each line as a number of asteriks.
            // Actually line i has n-i+1 asterisks
        {
            Console.Write("* ");
        }

        // jump and get ready for the next line
        Console.WriteLine();
    }
}
```

Exercise 11

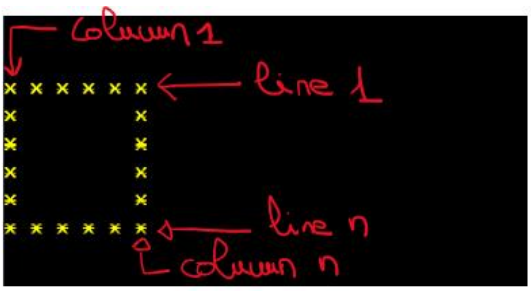


TOP is n asteriks ⇒ repeat n times write *
MIDDLE is n-2 lines ⇒ repeat n-2 times Write a line
BOTTOM is n asterisks ⇒ same as top

```
// TOP (just n asteriks)
for (int ast=1; ast<=n; ast++)
{
    Console.Write("* ");
}
Console.WriteLine();

// MIDDLE PART
// n-2 lines. Each line is *, n-2 blank spaces, *
for (int line=1; line<=n-2; line++)
{
    Console.Write("* "); // first asterisks
    // now the n-2 blank spaces
    for (int blank = 1; blank<=n-2; blank++)
    {
        Console.Write("  ");
    }
    Console.Write("* "); // the last asterik
    Console.WriteLine();
}

// BOTTOM (just n asteriks, like the top)
for (int ast = 1; ast <= n; ast++)
{
    Console.Write("* ");
}
```

A different approach is shown on next page

Exercise 11 -second approach-



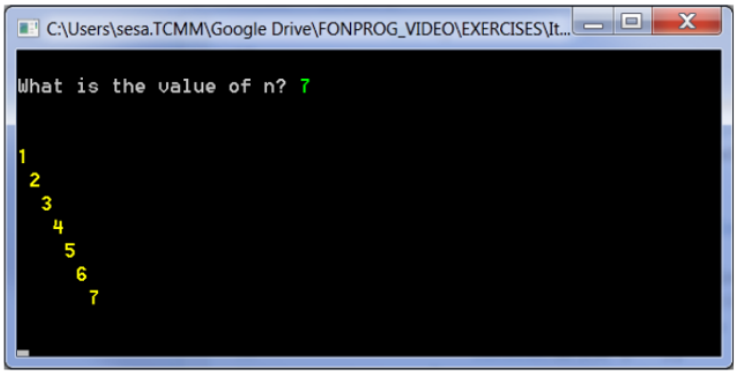

```
/*
 The square has n lines.
 Each line has n characters.
 The character is * when
    - line is the first
    - line is the last
    - column is the first
    - column is the last
*/

for (int line = 1; line <= n; line++) // iterate for n lines
{
    for (int column = 1; column <= n; column++) // iterate for n columns
    {
        if (line==1 || line==n || column==1 || column==n)
        {
            Console.Write("* ");
        }
        else
        {
            Console.Write("  ");
        }
    }
    Console.WriteLine();
}
```

Exercise 12

Ideas similar to those applied to the previous exercise can be applied here:

- Approach 1: divide the triangle into three sections:
  - Top: just one asterisk
  - Middle: n-2 lines with an asterisk, some blank spaces, an asterisk
  - Bottom line: n asterisk
- Approach 2: in some places there's an asterisk, in some others there's a blank space
  - An asterisk when line==n; when column==1 or when column==line
  - A blank space otherwise

Exercise 13



Each line is
* some blank spaces
* the number of the line

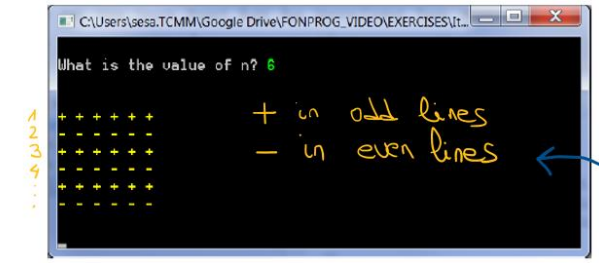
HOW MANY BLANK SPACES ARE THERE IN EACH LINE? line-1

```
for (int line = 1; line <=n; line++)
{
    // each line has line-1 spaces before the number
    for (int spaces = 1; spaces<=line-1; spaces++)
    {
        Console.Write(" ");
    }

    // the number of the line and then jump
    Console.WriteLine(line);
}
```

Exercise 14

Each line has n-line blank spaces preceding the number of the line

Exercise 15



repeat n times
write a line → repeat n times
write a character
the question is: which character ?

```
for (int line = 1; line<=n; line++)
    // an nxn square has n lines
{
    for (int charac = 1; charac<=n; charac++)
        // each line has exactly n characters
    {
        // the character depends on the line:
        // + for odd lines
        // - for even lines
        if (line%2==0)
        {
            Console.Write("- ");
        }
        else
        {
            Console.Write("+ ");
        }
    }

    Console.WriteLine();
}
```

for each character
make a decision

A different approach could be...

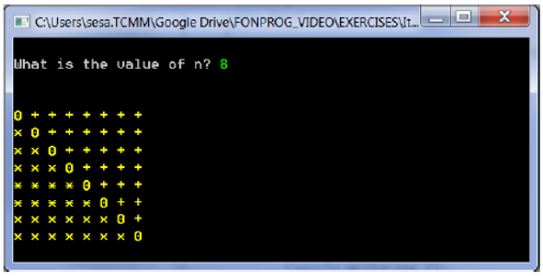
```
for (int line = 1; line<=n; line++)
    // an nxn square has n lines
{
    // depending on the parity the line is made of
    // + characters or - characters
    if (line%2==0)
    {
        for (int charac=1; charac<=n; charac++)
        {
            Console.Write("- ");
        }
    }
    else
    {
        for (int charac = 1; charac <= n; charac++)
        {
            Console.Write("+ ");
        }
    }

    Console.WriteLine();
}
```

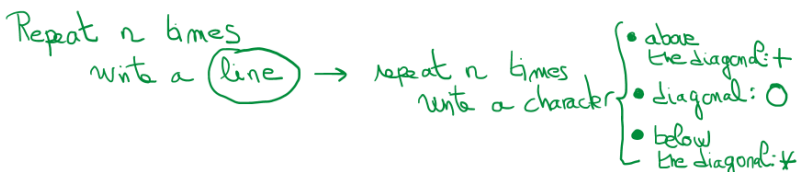
make the decision
at the line level

Exercise 16

For each line…
For each column in the line
Write the number of the line when column==line; write an asterisk otherwise

Exercise 17





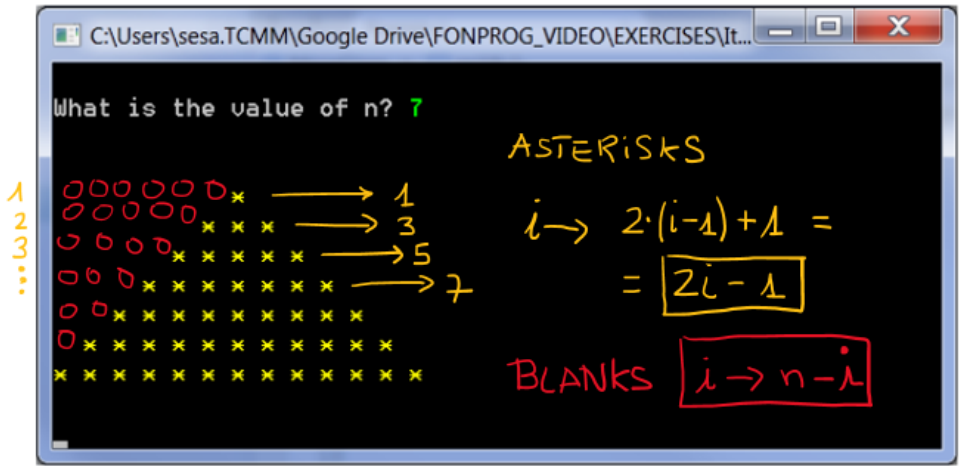

```
// as usual, an nxn squere has n lines...
for (int line=1; line<=n; line++)
{
    // and each line has n columns...
    for (int column = 1; column<=n; column++)
    {
        // the character depends on the relationship between
        // line and column...
        if (line==column) { Console.Write("0 ");}  // diagonal
        else if (line>column) { Console.Write("* ");}  // below the diagonal
        else { Console.Write("+ "); }  // above the diagonal
    }

    Console.WriteLine();
}
```

A different approach…

```
// as usual, an nxn squere has n lines...
for (int i=1; i<=n; i++)
{

    // each line has some asterisks, 0, some pluses
    /*
        line 1 has 0 asterisks, 0, n-1 pluses
        line 2 has 1 asterisk, 0, n-2 pluses
        line 3 has 2 asteriks, 0, n-3 pluses
        ...
        line i has i-1 asterisks, 0, n-i pluses
     */

    // the i-1 asterisks
    for (int ast=1; ast<=i-1; ast++) { Console.Write("* "); }

    // the 0
    Console.Write("0 ");

    // the n-i pluses
    for (int plu = 1; plu <= n-i; plu++) { Console.Write("+ "); }

    Console.WriteLine();
}
```

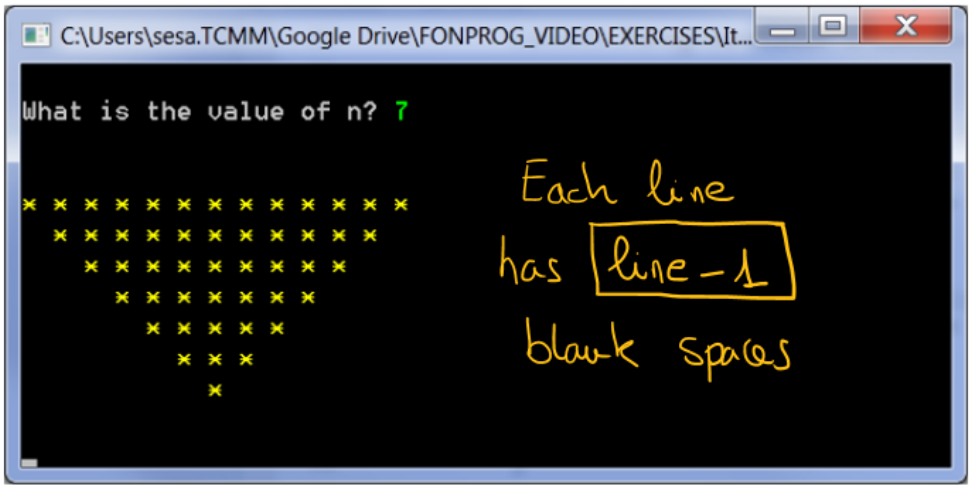
Exercise 18

Each line has some blanks and then some asterisk. The clue is to determine the number of blanks and the number of asterisk in each line as a function of the number of the line …




```
for (int line = 1; line <=n; line++)
{
    // each line has...

    // ... n-line blank spaces...
    for(int blank = 1; blank<=n-line; blank++)
    {
        Console.Write("  ");
    }

    // ... and then 2*line-1 asteriks
    for (int ast = 1; ast<=2*line-1; ast++)
    {
        Console.Write("* ");
    }

    Console.WriteLine();
}
```

Exercise 19




```
for (int line=1; line<=n; line++)
{
    // first line-1 blank spaces...
    for (int blank=1; blank<=line-1; blank++)
    {
        Console.Write("  ");
    }

    // ... then 2(n-line)+1 asteriks
    for (int ast = 1; ast <= 2 * (n - line) + 1; ast++)
    {
        Console.Write("* ");
    }

    Console.WriteLine();
}
```

While-based iterations: finding the right condition

In the following exercises you have to write while-based iterations to achieve the specified behaviour. All random numbers must be in the interval [1,6] –as if simulating the rolling of a die. Images show possible (correct) outcomes of the programs.

Most of them have a skeleton like this:

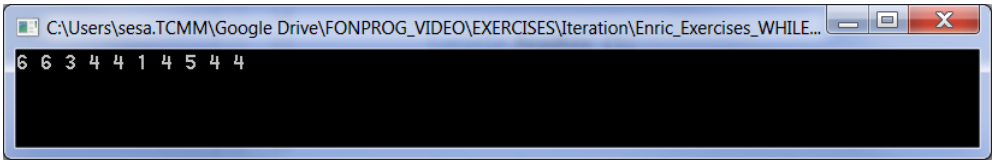
```
public static void Main (string[] args)
{
    int number;
    Random alea = new Random ();
    // declare more variables if needed

    // make the necessary initializations (at least of the
    // variables that appear in the condition)
    // Sometimes a first step is made here

    while (/* COMPLETE */) {
        // the condition governing the while is just
        // the negation of the goal

        /* COMPLETE */
        // inside the while make one step towards
        // achieving the goal
    }
    // GOAL: what happens when this point is reached?
    // (what condition makes the iteration stop?)

    Console.ReadKey ();
}
```

This skeleton fits well in any of the iterative patterns seen in class.

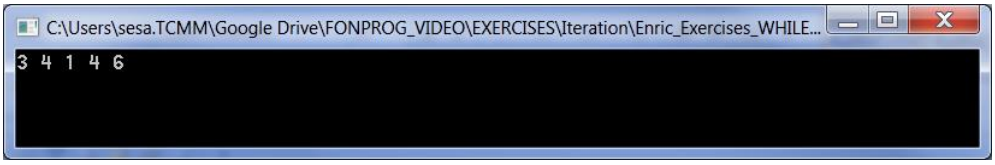
The most important part of these exercises if finding the (right) condition that governs the iteration. In all cases, you are advised to think in terms of the goal: what must happen when the iteration finishes (what makes the iteration finish). Then, the governing condition is just its negation

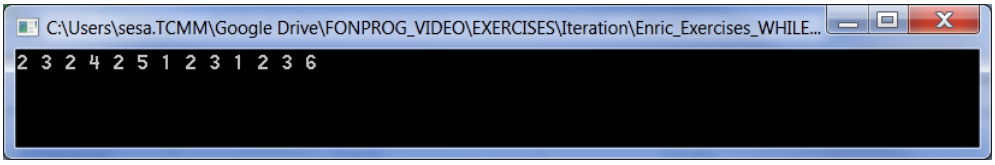
Solutions are provided at the end of the section

21. Generate and write 10 random numbers

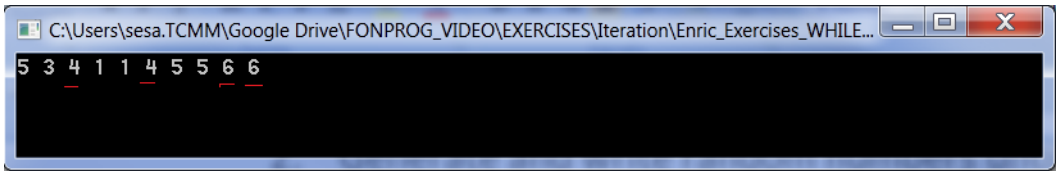


22. Generate and write random numbers until a six is written

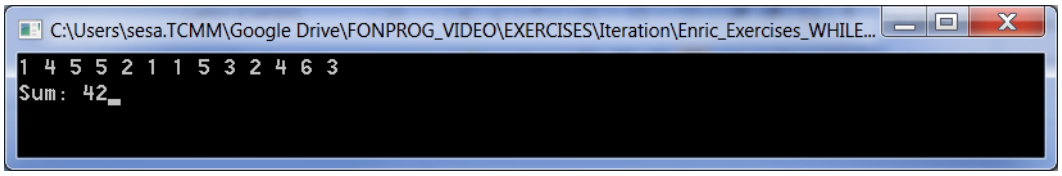




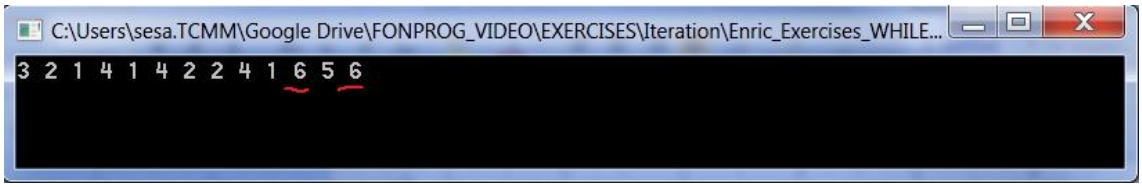
23. Generate and write random numbers until four even numbers have been written



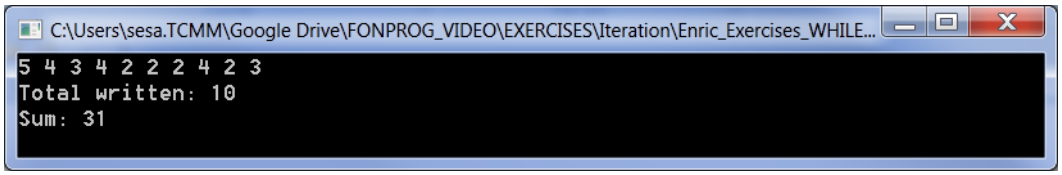
24. Generate and write random numbers until they add up to **more** than forty

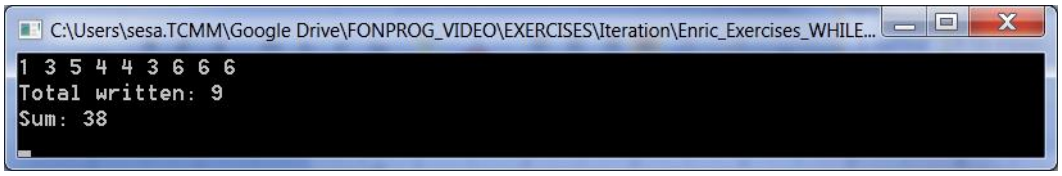


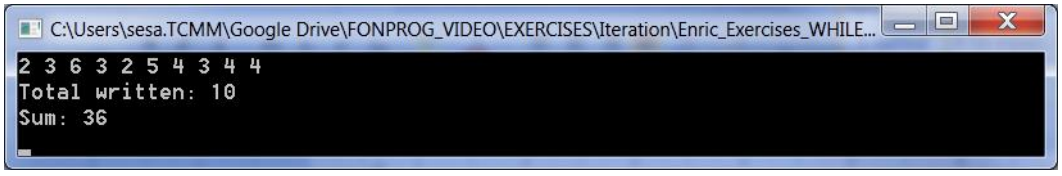
25. Generate and write random numbers until two sixes have been written



26. Generate and write random numbers until they add up to more than thirty-five or exactly ten numbers have been written

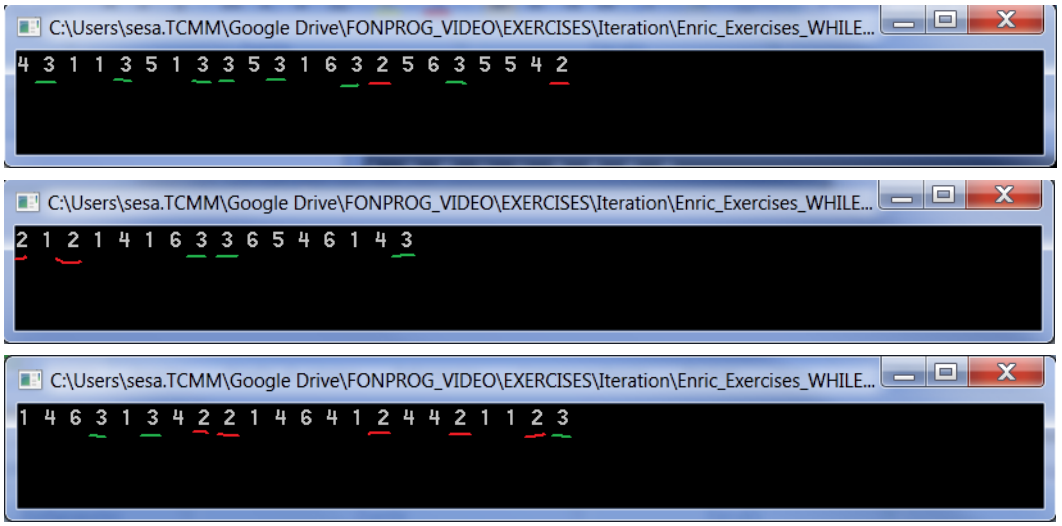




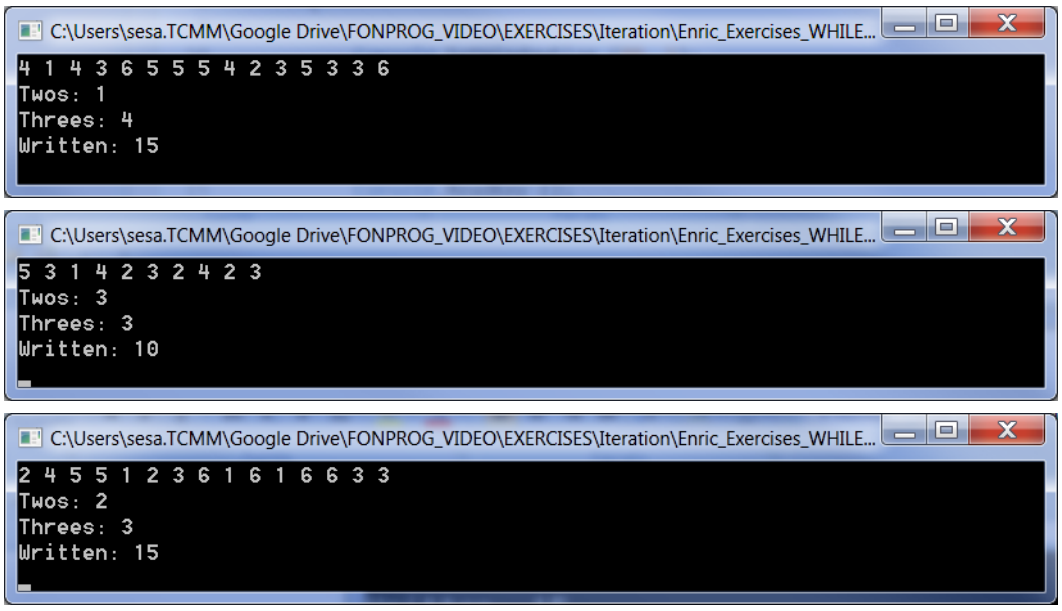


27. Generate and write random numbers until **at least** two two's **and** three three's have been written

```
int number;
Random alea = new Random ();
int twos, threes;

twos = 0;
threes = 0;
while (/* COMPLETE */) {
    number = alea.Next (1, 7);
    Console.Write (number + " ");

    /* COMPLETE */
}
```



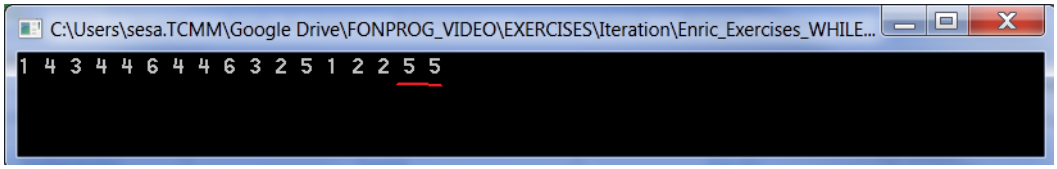
28. Generate and write random numbers until **at least** two two's and three three's have been written **or** a maximum of fifteen numbers have been written



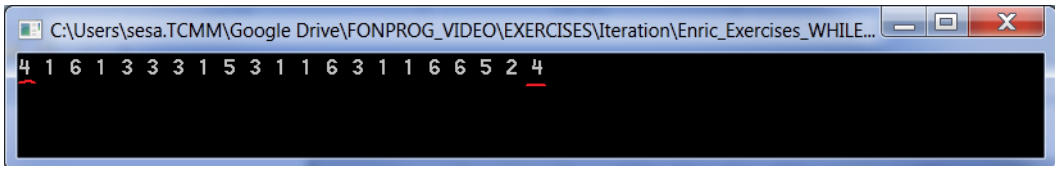
29. Generate and write random numbers until two consecutive fives have been written

```
Random alea = new Random ();
int prior, current;

prior = alea.Next (1, 7);
current = alea.Next (1, 7);
Console.Write (prior + " " + current + " ");

while (/* COMPLETE */) {
    // you can only roll one die here...
    /* COMPLETE */

    Console.Write (current + " ");
}
```

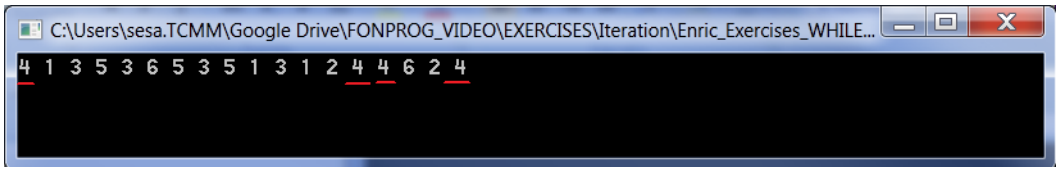
> At the very beginning the die is cast twice From that point on, only one die is cast every time. Then that die is the current, and the previous current becomes the prior.
> **This is <u>not</u> equivalent to casting pairs of dice until a double five is got**



30. Generate and write random numbers until the first is repeated.
    Hint: the first die must be cast outside the iteration and kept in a separate variable… Then cast the current and keep on casting it inside the iteration…

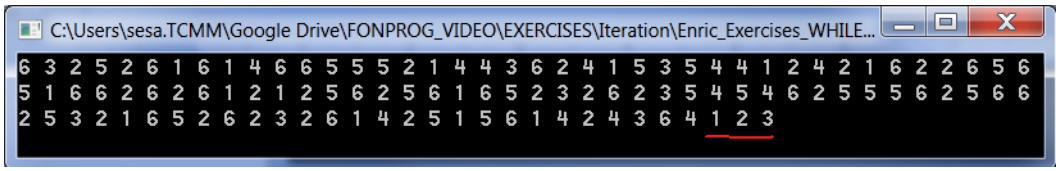


31. Generate and write random numbers until the first has been repeated three more times.

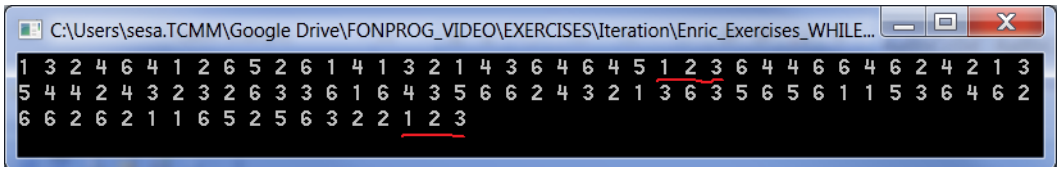


32. Generate and write random numbers until the sequence 1 2 3 has been written

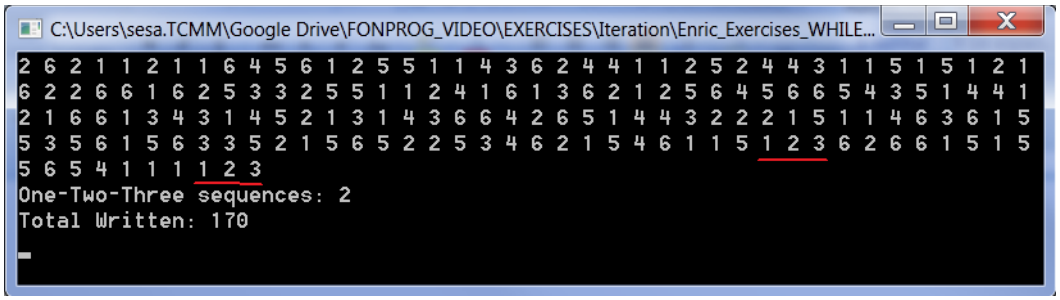
> At the very beginning the die is cast three times. From then on, it's cast once in each iteration.
> **This is <u>not</u> equivalent to casting three dice simultaneously until a 1 2 3 shows up**

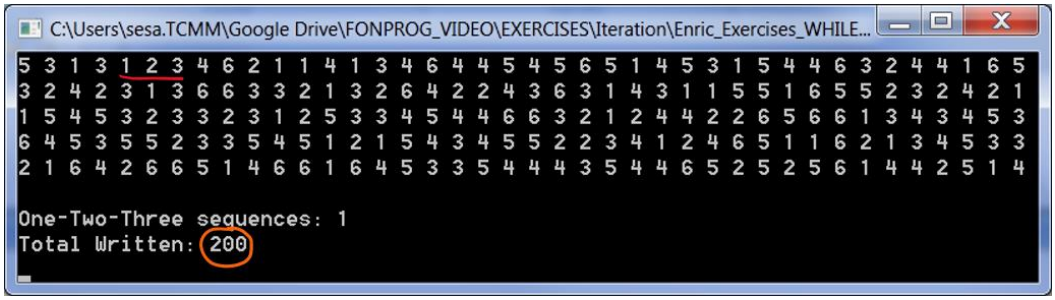


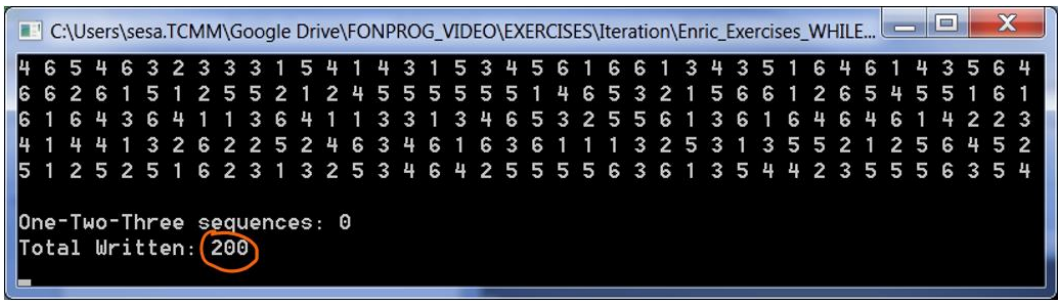
33. Generate and write random numbers until the sequence 1 2 3 has been written twice



34. Generate and write random numbers until the sequence 1 2 3 has been written twice **or** a maximum of two hundred numbers have been written







```
// Exercise 21. SOLUTION
public static void Main (string[] args)
{
    int number;
    Random alea = new Random ();
    // declare more variables if needed
    int counter;

    // generate and write 10 random numbers
    counter = 0;
    while (!(counter==10)) {
        // condition could also be counter!=10
        number = alea.Next (1, 7);
        Console.Write (number + " ");
        counter++;
    }
    // This point is reached because counter==10

    Console.ReadKey ();
```

```
// Exercise 22. Solution
public static void Main (string[] args)
{
    int number;
    Random alea = new Random ();

    // generate and write random numbers until a six is written
    number = alea.Next (1, 7);
    Console.Write (number + " ");
    while (!(number == 6)) {
        // condition could also be number!=6
        number = alea.Next (1, 7);
        Console.Write (number + " ");
    }
    // when this point is reached
    // number==6

    Console.ReadKey ();
}
```

```
// Exercise 23. Solution
public static void Main (string[] args)
{
    int number;
    Random alea = new Random ();
    int counter;

    // generate and write random numbers until
    // four even numbers have been written
    counter = 0;
    while (!(counter==4)) {
        // condition could also be counter!=4 or counter<4
        number = alea.Next (1, 7);
        Console.Write (number + " ");
        if (number % 2 == 0) {
            counter++;
        }
    }
    // This point is reached because
    // counter==4

    Console.ReadKey ();
}
```

```
// Exercise 24. Solution
public static void Main (string[] args)
{
    int number;
    Random alea = new Random ();
    // declare more variables if needed
    int sum;

    // generate and write random numbers until
    // they add up to more than forty
    sum = 0;
    while (sum <= 40) {
        // notice that !(sum>40) is equivalent to sum<=40
        number = alea.Next (1, 7);
        Console.Write (number + " ");
        sum = sum + number;
    }
    // Goal: sum>40

    Console.Write ("\nSum: " + sum);

    Console.ReadKey ();
}
```

```
// Exercise 25. Solution
public static void Main (string[] args)
{

    int number;
    Random alea = new Random ();
    int counter;

    // generate and write random numbers
    // until two sixs (6) have been written

    counter = 0;

    while (!(counter==2)) {
        number = alea.Next (1, 7);
        Console.Write (number + " ");
        if (number == 6) {counter++;}
    }
    // GOAL: counter == 2

    Console.ReadKey ();
}
```

```
// Exercise 26. Solution
public static void Main (string[] args)
{
    ...

    sum = 0;
    written = 0;
    while (!(sum > 35 || written == 10)) {
        // (sum <= 35 && written < 10) also correct
        number = alea.Next (1, 7);
        Console.Write (number + " ");
        sum = sum + number;
        written++;
    }
    // GOAL: sum>35 || written==10

    Console.WriteLine ("\nTotal written: " + written);
    Console.WriteLine ("Sum: " + sum);

    Console.ReadKey ();
}
```

// exercise 27

```
twos = 0;
threes = 0;

while (!(twos >= 2 && threes >= 3)) {
    number = alea.Next (1, 7);
    Console.Write (number + " ");
    if (number == 2) { twos++;}
    else if (number == 3) {threes++;}
}

// GOAL: at least two twos and and least three threes
// GOAL: twos >= 2 && threes >= 3
```

Applying pattern #3…

```
twos = 0;
threes = 0;
successful = false;

while (!successful) {
    number = alea.Next (1, 7);
    Console.Write (number + " ");
    if (number == 2) { twos++;}
    else if (number == 3) {threes++;}

    successful = (twos >= 2 && threes >= 3);
}

// GOAL: at least two twos and and least three threes
// GOAL: twos >= 2 && threes >= 3
// GOAL: also successful == true
```

// exercise 28

```
twos = 0;
threes = 0;
written = 0;

while (!((twos >= 2 && threes >= 3) || written==15)) {
    number = alea.Next(1, 7);
    Console.Write(number + " ");
    if (number == 2) { twos++; }
    else if (number == 3) { threes++; }
    written++;
}

// WHY HERE?
//      at least two twos and and least three threes
//      OR...
//      15 numbers written
// (twos >= 2 && threes >= 3) || written==15
```

```
// Exercise 29. Solution
public static void Main (string[] args)
{
    Random alea = new Random ();
    int prior, current;

    // generate and write random numbers
    // until two consecutive fives have been written

    prior = alea.Next (1, 7);
    current = alea.Next (1, 7);
    Console.Write (prior + " " + current + " ");

    while (!(prior == 5 && current == 5)) {
        prior = current;
        current = alea.Next (1, 7);
        Console.Write (current + " ");
    }
    // GOAL: the prior die and the current one are 5
    // prior==5 && current==5

    Console.ReadKey ();
}
```

```
// Exercise 30. Solution
public static void Main (string[] args)
{
    Random alea = new Random ();
    int first, current;

    // generate and write random numbers
    // until the first is repeated

    first = alea.Next (1, 7);
    current = alea.Next (1, 7);
    Console.Write (first + " " + current + " ");

    while (!(current==first)) {
        // (current!=first) would be correct too
        current = alea.Next (1, 7);
        Console.Write (current + " ");
    }
    // GOAL: current == first

    Console.ReadKey ();
}
```

```
// Exercise 31. Solution
public static void Main (string[] args)
{
    Random alea = new Random ();
    int first, current;
    int counter;

    first = alea.Next (1, 7);
    Console.Write (first + " ");
    counter = 0;

    while (!(counter == 3)) {
        current = alea.Next (1, 7);
        Console.Write (current + " ");
        if (current == first) {
            counter++;
        }
    }
    // GOAL: counter==3

    Console.ReadKey ();
}
```

```
// Exercise 32. Solution
public static void Main (string[] args)
{
    Random alea = new Random ();
    int first, second, third;

    first = alea.Next (1, 7);
    second = alea.Next (1, 7);
    third = alea.Next (1, 7);
    Console.Write (first + " " + second + " " + third + " ");

    while (!(first == 1 && second == 2 && third == 3)) {
        first = second;
        second = third;
        third = alea.Next (1, 7);
        Console.Write (third + " ");
    }
    // GOAL: first==1 and second==2 and third==3

    Console.ReadKey ();
}
```

```
// Exercise 33. solution
public static void Main (string[] args)
{
    Random alea = new Random ();
    int first, second, third;
    int counter;

    // generate and write random numbers
    // until the sequence 1 2 3 has been written twice

    first = alea.Next (1, 7);
    second = alea.Next (1, 7);
    third = alea.Next (1, 7);
    Console.Write (first + " " + second + " " + third + " ");

    if (first == 1 && second == 2 && third == 3) {
        counter = 1;
    } else {
        counter = 0;
    }

    while (!(counter==2)) {
        first = second;
        second = third;
        third = alea.Next (1, 7);
        Console.Write (third + " ");
        if (first == 1 && second == 2 && third == 3) {
            counter++;
        }
    }
    // GOAL: counter==2

    Console.ReadKey ();
}
```

```
// Exercise 34. Solution
public static void Main (string[] args)
{
    Random alea = new Random ();
    int counter, written, first, second, third;

    first = alea.Next (1, 7);
    second = alea.Next (1, 7);
    third = alea.Next (1, 7);

    Console.Write (first + " " + second + " " + third + " ");
    if (first == 1 && second == 2 && third == 3) {
        counter = 1;
    } else {
        counter = 0;
    }
    written = 3;

    while (!(counter==2 || written==200)) {
        // (counter < 2 && written < 200) would be correct too
        first = second;
        second = third;
        third = alea.Next (1, 7);
        Console.Write (third + " ");
        if (first == 1 && second == 2 && third == 3) {
            counter++;
        }
        written++;
    }
    // Why here?  counter==2 or written==200

    Console.WriteLine ("\nOne-Two-Three sequences: " + counter);
    Console.WriteLine ("Total Written: " + written);

    Console.ReadKey ();
}
```

"Advanced" exercises

In the following exercises, use the type of iteration (for or while) that you deem more appropriate. No solutions are provided.

35. Write a program that given a positive integer (>0) counts how many divisors it has, including 1 and itself. If the number is not positive, the program "complains" and terminates. **Hint**: the divisors of any positive number n are all located in the interval [1,n].

```
Please enter a positive integer (>0) -3
Incorrect number -3
Press any key to exit
```

```
Please enter a positive integer (>0) 15
Number 15 has 4 divisors
Press any key to exit
```

```
Please enter a positive integer (>0) 24
Number 24 has 8 divisors
Press any key to exit
```

```
Please enter a positive integer (>0) 17
Number 17 has 2 divisors
Press any key to exit
```

36. Same as before but instead of counting the divisors, the program prints them

```
Please enter a positive integer (>0) 24
The divisors of 24 are:
   1
   2
   3
   4
   6
   8
   12
   24
Press any key to exit
```

```
Please enter a positive integer (>0) 17
The divisors of 17 are:
   1
   17
Press any key to exit
```

37. Write a program that given a positive integer (>0) says whether it is prime or not. Remember: a prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself. Also remember that 1 is not a prime number. If the number is not positive the program "complains" and terminates.

```
Please enter a positive integer (>0) 1
Number 1 IS NOT PRIME
Press any key to exit
```

```
Please enter a positive integer (>0) 2
Number 2 IS PRIME
Press any key to exit
```

```
Please enter a positive integer (>0) 1093
Number 1093 IS PRIME
Press any key to exit
```

```
Please enter a positive integer (>0) 2331
Number 2331 IS NOT PRIME
Press any key to exit
```

38. Write a program that given a positive integer (>0) prints its prime divisors. If the number is not positive the program “complains” and terminates. Hint: locate divisors. Once a divisor has been located, check whether it is prime or not.

```
Please enter a positive integer (>0) 2331

The prime divisors of 2331 are:
   3
   7
   37

Press any key to exit
```

```
Please enter a positive integer (>0) 17

The prime divisors of 17 are:
   17

Press any key to exit
```

```
Please enter a positive integer (>0) 1

The prime divisors of 1 are:

Press any key to exit _
```

```
Please enter a positive integer (>0) 105675570

The prime divisors of 105675570 are:
   2
   3
   5
   7
   11
   13
   17
   23

Press any key to exit _
```

39. Write a program that given a positive integer (>0) counts its digits. If the number is not positive the program “complains” and terminates. Hint: Divide into 10 until result is 0. Count how many times you can divide.

```
Please enter a positive integer (>0) 7

The number 7 has 1 digit(s)

Press any key to exit
```

```
Please enter a positive integer (>0) 1234

The number 1234 has 4 digit(s)

Press any key to exit
```

40. Write a program that given a positive integer (>0) prints the list of its digits (starting by the units). If the number is not positive the program “complains” and terminates.

```
Please enter a positive integer (>0) 7

these are the digits of 7 (in reverse order)
  7

Press any key to exit
```

```
Please enter a positive integer (>0) 128544

these are the digits of 128544 (in reverse order)
  4  4  5  8  2  1

Press any key to exit
```

41. Write a program that given a double number (the base) and a positive (>=0) int number (the exponent) computes **base**$^{\text{exponent}}$. If exponent is negative, the program complains and terminates.

```
Please enter the base 2.5
Please enter the exponent (integer and >=0) 0

2.5 to the 0-th power is 1

Press any key to exit
```

```
Please enter the base 7.9
Please enter the exponent (integer and >=0) 1

7.9 to the 1-st power is 7.9

Press any key to exit
```

```
Please enter the base -4
Please enter the exponent (integer and >=0) 2

-4 to the 2-nd power is 16

Press any key to exit
```

```
Please enter the base 8
Please enter the exponent (integer and >=0) 3

8 to the 3-rd power is 512

Press any key to exit
```

```
Please enter the base -2
Please enter the exponent (integer and >=0) 7

-2 to the 7-th power is -128

Press any key to exit
```

```
Please enter the base 2.5
Please enter the exponent (integer and >=0) -6

Negative numbers are not allowed


Press any key to exit
```

42. Write a program that given a positive integer (>=0) prints its factorial. Remember that n! = 1 · 2 · 3 · … · (n-1) · n and that 0! = 1. If the number is negative the program complains and terminates

```
Please enter a positive integer (>=0) 0

The factorial of 0 is 1

Press any key to exit
```

```
Please enter a positive integer (>=0) 1

The factorial of 1 is 1

Press any key to exit
```

```
Please enter a positive integer (>=0) 5

The factorial of 5 is 120

Press any key to exit
```

```
Please enter a positive integer (>=0) 10

The factorial of 10 is 3628800

Press any key to exit
```

```
Please enter a positive integer (>=0) -12

Sorry, can't compute the factorial of a negative number

Press any key to exit
```

43. Write a program that prints the numbers in the Fibonacci sequence up to the i-th term. The number i (i>=1) will be provided by the user.

$$f(n) = \begin{cases} 0 & if \quad n = 0 \\ 1 & if \quad n = 1 \\ F(n-1) + F(n-2) & if \quad n > 1 \end{cases}$$

```
Please enter the index of the last term (i>=1) 1

  F(0) = 0
  F(1) = 1

Press any key to exit
```

```
Please enter the index of the last term (i>=1) 2

  F(0) = 0
  F(1) = 1
  F(2) = 1

Press any key to exit
```

```
Please enter the index of the last term (i>=1) 7

  F(0) = 0
  F(1) = 1
  F(2) = 1
  F(3) = 2
  F(4) = 3
  F(5) = 5
  F(6) = 8
  F(7) = 13

Press any key to exit
```

```
Please enter the index of the last term (i>=1) 19

  F(0) = 0
  F(1) = 1
  F(2) = 1
  F(3) = 2
  F(4) = 3
  F(5) = 5
  F(6) = 8
  F(7) = 13
  F(8) = 21
  F(9) = 34
  F(10) = 55
  F(11) = 89
  F(12) = 144
  F(13) = 233
  F(14) = 377
  F(15) = 610
  F(16) = 987
  F(17) = 1597
  F(18) = 2584
  F(19) = 4181

Press any key to exit
```

OTHER EXERCISES

Exercise “Highest card”

An incomplete program to play (iterated) Highest Card is provided. Crucial parts of the program, related to its iterative behaviour, are missing. Complete them.

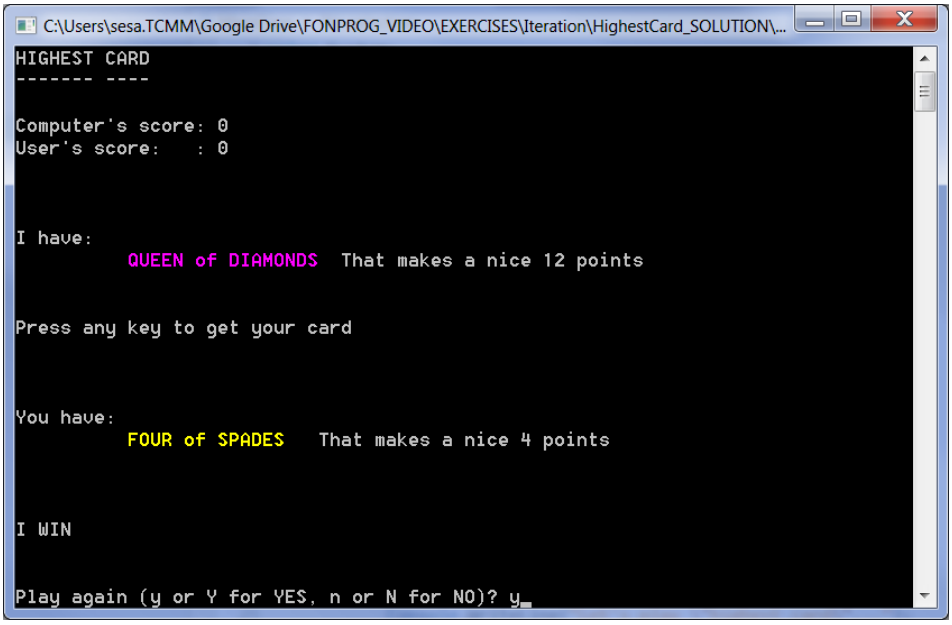
```
public static void Main (string[] args)
{

    string suitName, cardName;
    bool wannaPlay;
    int computerSuit, userSuit;
    int computerNumber, userNumber;
    int computerScore, userScore;
    int totalComputerScore, totalUserScore;
    char answer;
    Random alea = new Random ();

    totalComputerScore = 0;
    totalUserScore = 0;

    // some introduction
    Console.WriteLine ("HIGHEST CARD");
    Console.WriteLine ("------- ----\n\n");
    Console.WriteLine("Let's play \"highest card\"...");
    Console.Write ("\n\nPress any key to start");
    Console.ReadKey ();

    wannaPlay = true;

    while (/*COMPLETE*/) {
        Console.Clear ();
        Console.WriteLine ("HIGHEST CARD");
        Console.WriteLine ("------- ----\n");
```

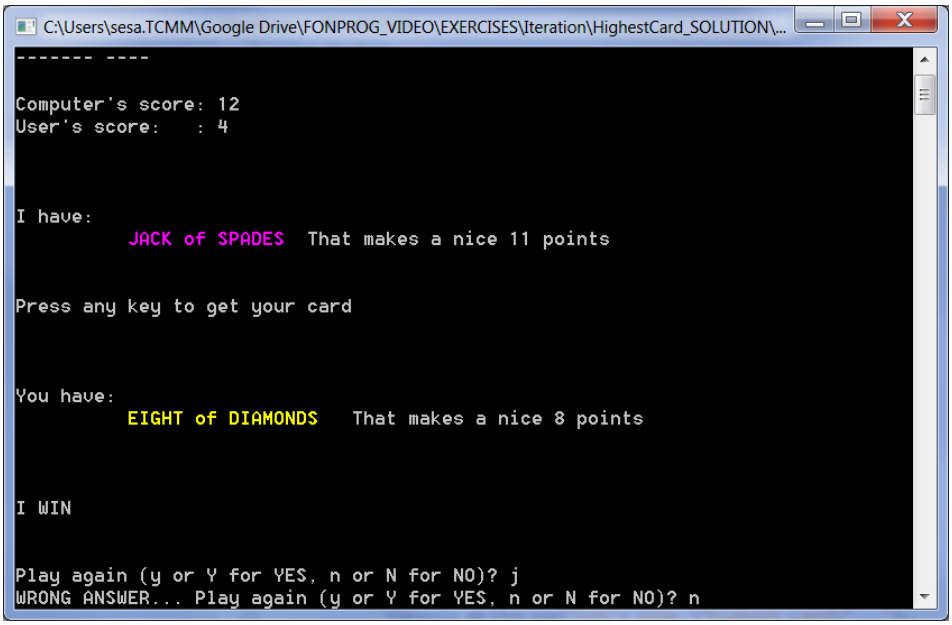
...

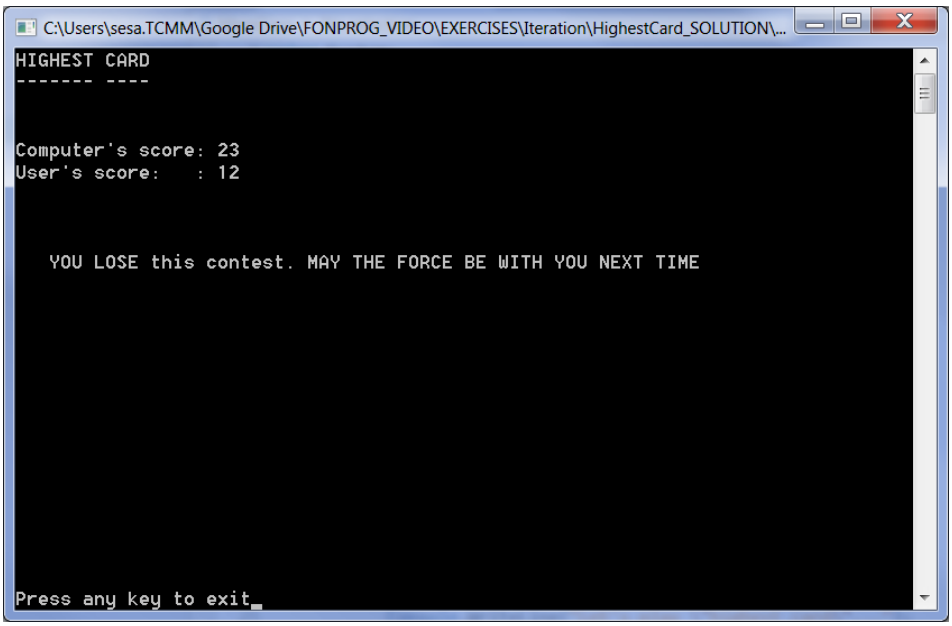
```
// generate user's card;
userSuit = alea.Next (1, 5);
userNumber = alea.Next (1, 14);
/*
    COMPLETE.
    DO NOT let the user get the same card as the computer. Iterate until the user gets a diferent card
*/
```

...

```
    // ask "play again?"
    /*
        COMPLETE.
        Correct answers are y, Y, n, or N
        If user gives an incorrect answer INSIST. Iterate unitl the answer is correct
    */
    wannaPlay = answer == 'y' || answer == 'Y';

} // end of while wanna play
```


C:\Users\sesa.TCMM\Google Drive\FONPROG_VIDEO\EXERCISES\Iteration\HighestCard_SOLUTION\...
HIGHEST CARD
------- ----
Computer's score: 0
User's score: : 0
I have:
QUEEN of DIAMONDS That makes a nice 12 points
Press any key to get your card
You have:
FOUR of SPADES That makes a nice 4 points
I WIN
Play again (y or Y for YES, n or N for NO)? y


C:\Users\sesa.TCMM\Google Drive\FONPROG_VIDEO\EXERCISES\Iteration\HighestCard_SOLUTION\...
------- ----
Computer's score: 12
User's score: : 4
I have:
JACK of SPADES That makes a nice 11 points
Press any key to get your card
You have:
EIGHT of DIAMONDS That makes a nice 8 points
I WIN
Play again (y or Y for YES, n or N for NO)? j
WRONG ANSWER... Play again (y or Y for YES, n or N for NO)? n

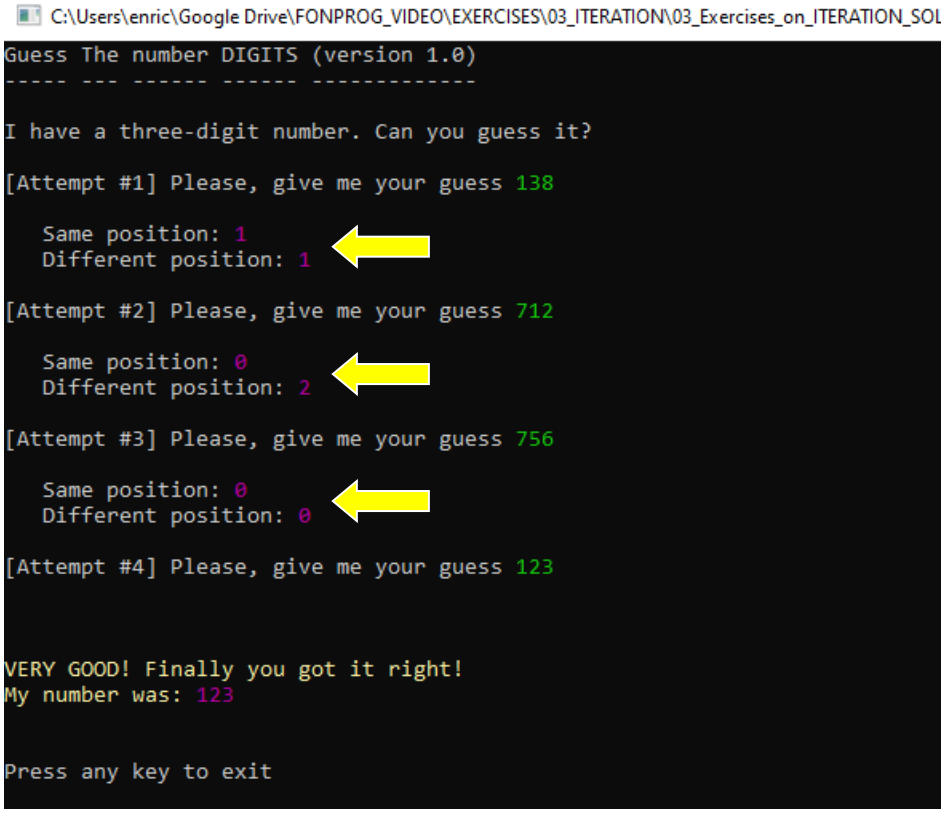

C:\Users\sesa.TCMM\Google Drive\FONPROG_VIDEO\EXERCISES\Iteration\HighestCard_SOLUTION\...
HIGHEST CARD
------- ----
Computer's score: 23
User's score: : 12
YOU LOSE this contest. MAY THE FORCE BE WITH YOU NEXT TIME
Press any key to exit

Exercise "Guess the number. Digits"

"Guess the number DIGITS" is a game in which the player has to guess a three-digit secret number randomly generated by the computer. After each incorrect guess, the program gives these clues:

1. The number of digits that not only appear in the secret number but also occupy the same position
2. The number of digits that appear in the secret number but not in the given positions

For instance, if the secret number were 123 and the user's guess were 138, the clues given by the program would be "digits in same position: 1; digits in different position: 1". If the user's guess had been 712 the clues would have been "digits in same position: 0; digits in different position: 2". Had the user's guess been 756 the clues would have been "digits in same position: 0; digits in different position: 0"



Several versions of the program implementing the game will have to be written, each one adding more features to the previous one.

VERSION 1.0

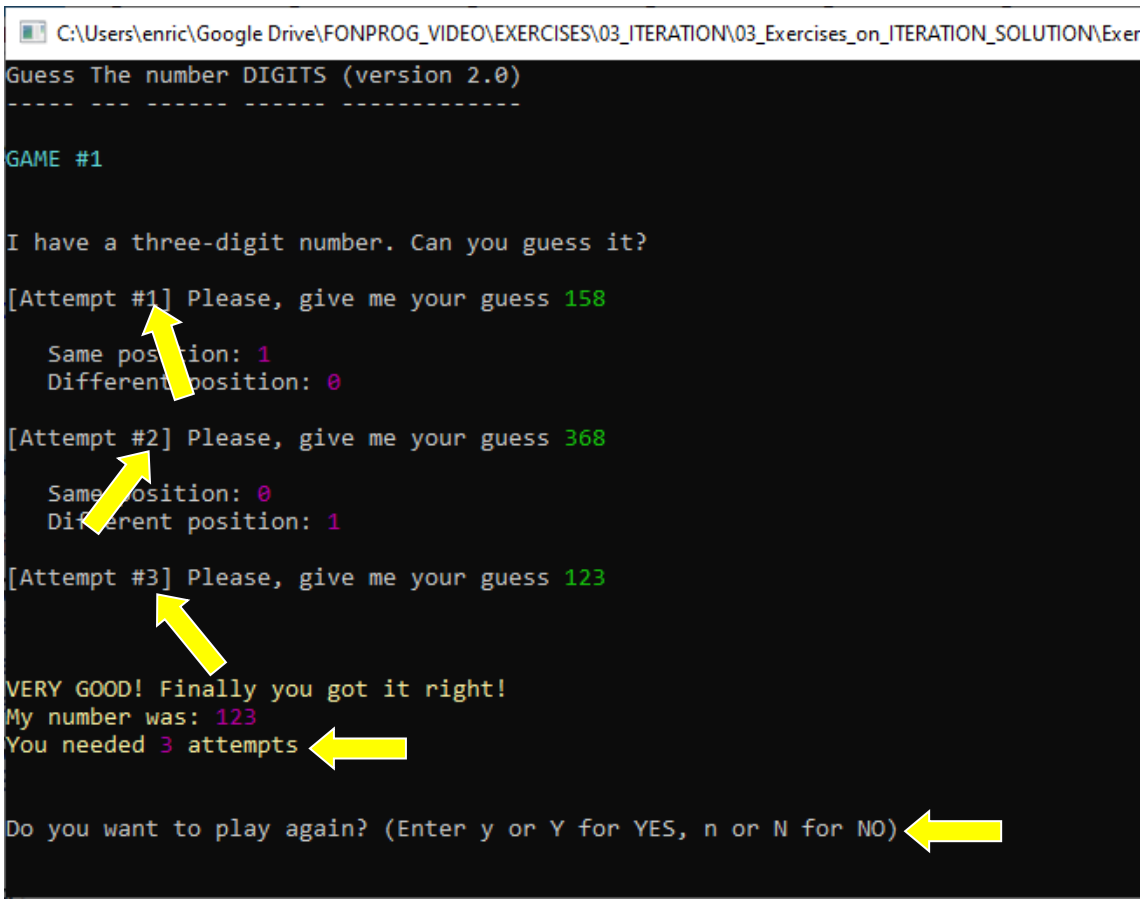
This is the simplest version. It is based on the following general structure

```
generate a random three-digit number
while (the game is not over) {
      get a guess from the user
      process the guess and determine if the game is over or not
}
```

VERSION 2.0

In this version, the following features are added to the previous one

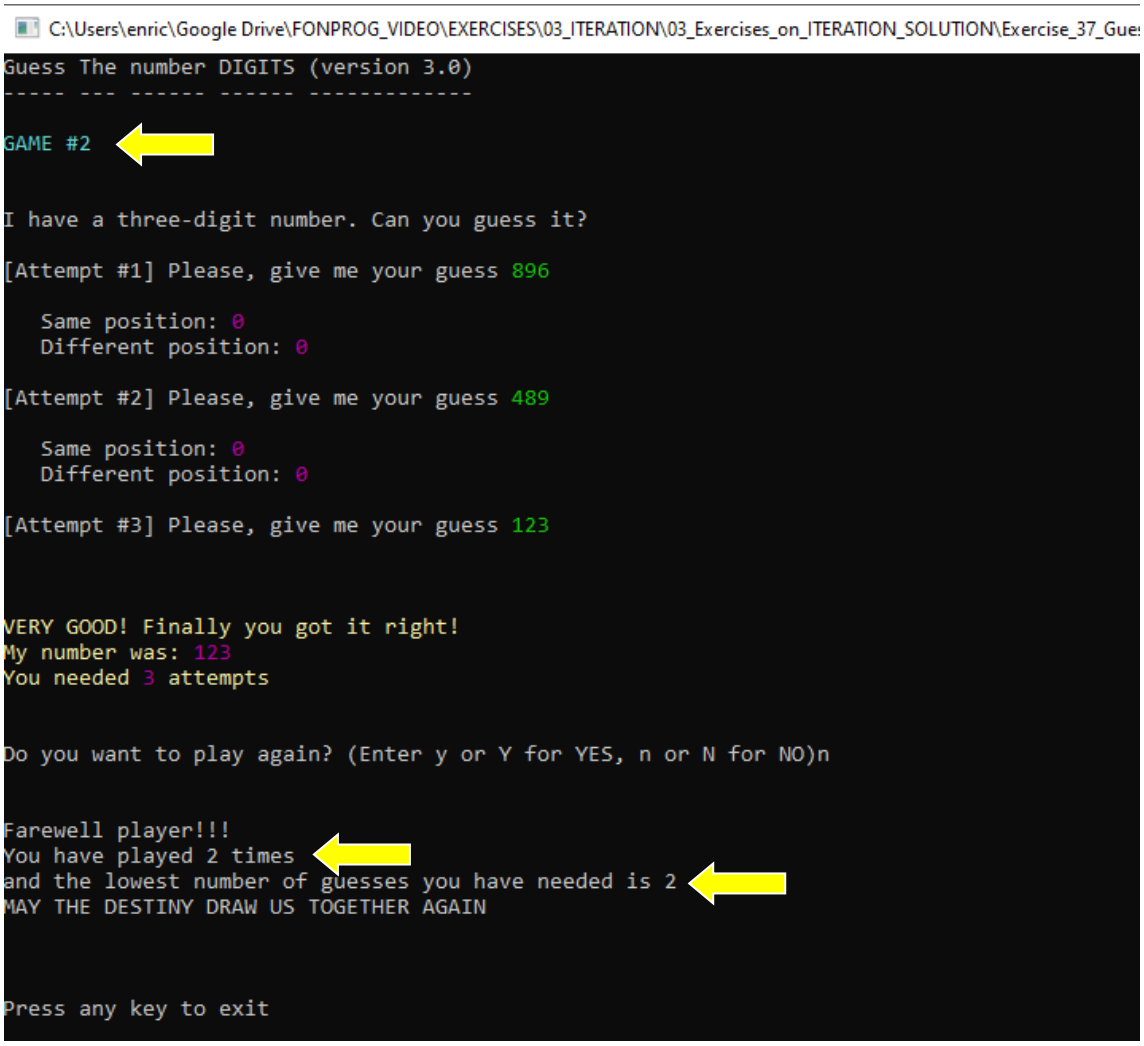
1. Attempts (user's guesses) are numbered
2. When the user has guessed the number, the total number of attempts is shown
3. When the user has guessed the number, the user is given the option of playing again (the program will generate a new random number).



VERSION 3.0

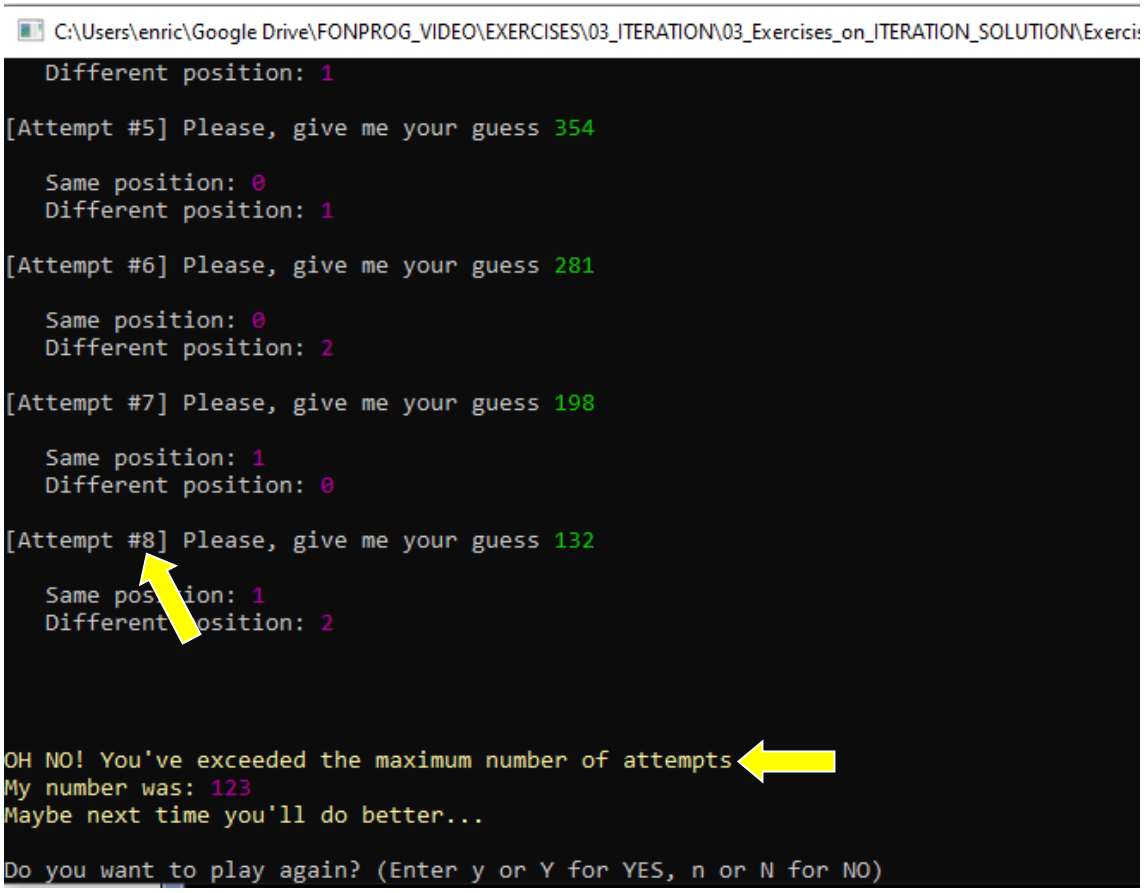
In this version:

1. Games are numbered
2. Once the user decides not to play again, the program shows:
    - The number of times the game has been played
    - The lowest number of attempts the user has needed to guess a number



VERSION 4.0

In the last version, the total number of attempts per number is limited to 8. After 8 unsuccessful attempts the game ends (though the user is given the option of playing again)



FUNCTIONAL DECOMPOSITION

PROCEDURES

"Reading" exercises

Solutions are given at the end of the section

1. Consider the following program (main and two function procedures)

```
public static void Main (string[] args)
{
    int number, a;

    Console.Write ("Please enter an integer number ");
    number = Convert.ToInt32 (Console.ReadLine());

    a = AddSomething (number);
    if (GoodForMe (number, a)) {
        a = AddSomething (a + 1);
    } else {
        number = AddSomething (a + AddSomething (number));
    }

    Console.WriteLine ("\nnumber = {0}, a = {1}", number, a);
}

static int AddSomething (int number) {
    if (number % 2 == 0) {
        number = number + 1;
    } else {
        number = number + 2;
    }
    return number;
}

static bool GoodForMe(int a, int b) {
    int remainderSum;

    remainderSum = a % 2 + b % 2;

    if (remainderSum == 0 || remainderSum == 2) {
        return true;
    } else {
        return false;
    }
}
```

(a) What will it write if the user input is 3?

(b) What will it write if the user input is 4?

2. Consider the following program (main and a function procedure)

```
public static void Main (String [] args) {

    int a, b, c;

    Console.Write ("Please enter an integer number ");
    a = Convert.ToInt32 (Console.ReadLine());
    Console.Write ("Please enter another integer number ");
    b = Convert.ToInt32 (Console.ReadLine());

    a = ShakeMeUp (b, a);

    b = ShakeMeUp (b, b);

    c = ShakeMeUp (ShakeMeUp (a, b), a);

    Console.WriteLine ("\na = {0}, b = {1}, c= {2}", a, b, c);
}

static int ShakeMeUp (int a, int b) {
    int c;

    if (a > b) {
        c = 2 * (a + b);
    }
    else {
        c = 2*(a-b);
    }
    return c-1;
}
```

(a) What will it write if the user input is 3 and 5?

(b) What will it write if the user input is 5 and 3?

3. Consider the following program (main and two function procedures)

```
public static void Main (String [] args) {
    int a, b;
    int final;

    Console.Write ("Please enter an integer number ");
    a = Convert.ToInt32 (Console.ReadLine());
    Console.Write ("Please enter another integer number ");
    b = Convert.ToInt32 (Console.ReadLine());

    final = SimpleOperation (a, b, SimpleQuestion (b, a));

    if (SimpleQuestion (final, b)) {

        final = final + SimpleOperation (b, final, false);

    } else {

        final = final - SimpleOperation (final, b, true);
    }

    Console.WriteLine ("\nfinal = {0}", final);
    Console.ReadKey ();
}


static int SimpleOperation (int x, int y, bool c) {
    if (c == SimpleQuestion (x, y)) {
        return x + y;
    } else {
        return y - x;
    }
}

static bool SimpleQuestion (int p, int q) {
    return p > 2 * q;
}
```

(a) What will it write if the user input is 4 and 15?

(b) What will it write if the user input is 15 and 4?

Solutions

1.

(a) What will it write if the user input is 3?
number = 3, a = 7

(b) What will it write if the user input is 4?
number = 11, a = 5

2.

(a) What will it write if the user input is 3 and 5?
a = 15, b = -1, c= 83

(b) What will it write if the user input is 5 and 3?
a = -5, b = -1, c= -9

3.

(a) What will it write if the user input is 4 and 15?
final = 7

(b) What will it write if the user input is 15 and 4?
final = -26

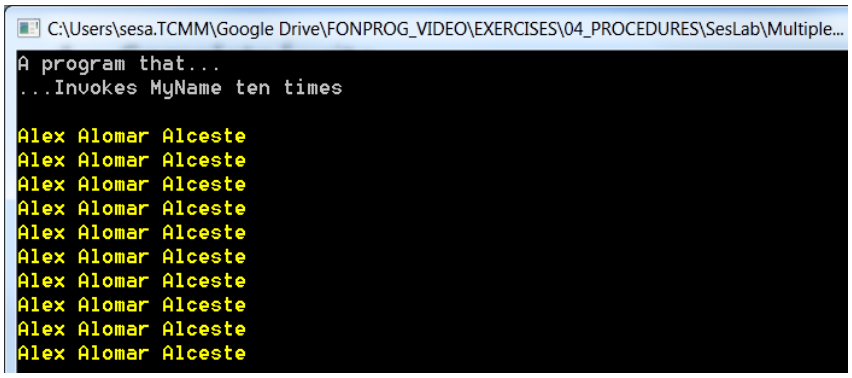
"Writing" exercises. FIRST STEPS

Solutions at the end of the section

4. Complete/write …

A. the parameterless action procedure MyName that writes your name in yellow. Notice that main already contains an invocation of this procedure.

```
public static void Main (string[] args)
{
        Console.WriteLine ("A program that...");
        Console.WriteLine ("...Invokes MyName ten times");
        Console.WriteLine ();

        for (int i = 1; i <= 10; i++) {
                MyName ();
        }

        Console.SetCursorPosition (0, Console.WindowHeight-1);
        Console.Write ("Press any key to exit");
        Console.ReadKey ();
}

// write here the parameterless action procedure myName that
// writes your name in yellow
```



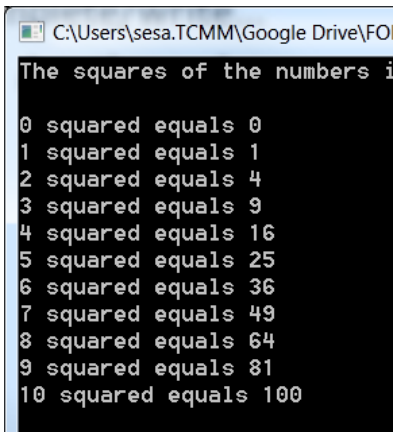
B. the one-parameter MyNameWithSpaces action procedure that writes your name in yellow but this time preceded by as many blank spaces as indicated in the parameter. Notice that main already contains an invocation of this procedure.

```
public static void Main (string[] args)
{
        Console.WriteLine ("A program that...");
        Console.WriteLine ("...Invokes MyNameWithSpaces ten times");
        Console.WriteLine ();

        for (int i = 1; i <= 10; i++) {
                MyNameWithSpaces (i);  // i is the number of spaces to
                                        //write in front of the name
        }

        Console.SetCursorPosition (0, Console.WindowHeight-1);
        Console.Write ("Press any key to exit");
        Console.ReadKey ();
}


// write here the one-parameter MyNameWithSpaces action procedure
// that writes your name in yellow but this time
// preceded by as many blank spaces as indicated in the parameter
```

5. Complete/write...

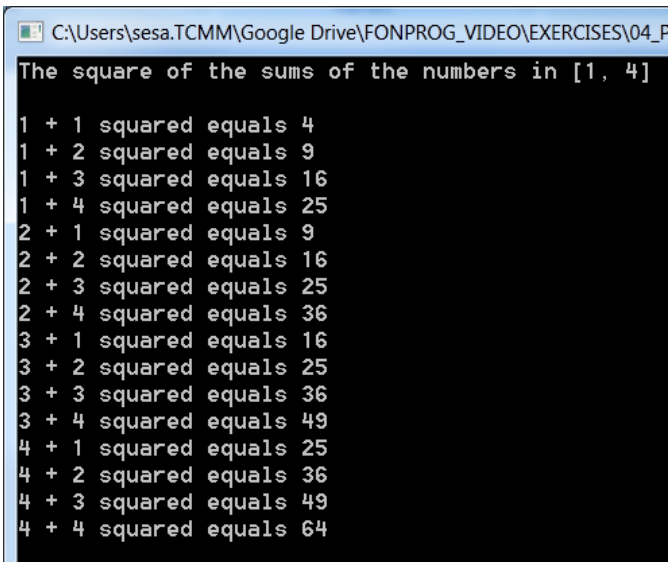
A. the code necessary for writing the squares of the numbers in [0, 10]. You must use the Square function already provided.

```
public static void Main (string[] args)
{
        // declare variables if needed...

        Console.WriteLine ("The squares of the numbers in [0, 10]");
        Console.WriteLine ();


        // write  here the code necessary for writing the squares of the
        // numbers in [0, 10]
        // you must use the Square function


        Console.SetCursorPosition (0, Console.WindowHeight-1);
        Console.Write ("Press any key to exit");
        Console.ReadKey ();
}

static int Square (int num) {
        // this function computes the square of the number given as parameter

        int result;

        result = num * num;

        return result;
}
```



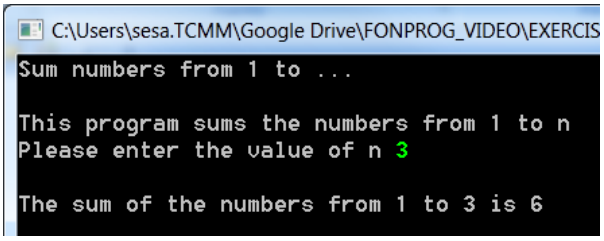
B. The function SquareOfTheSum that given two integer numbers computes the square of their sum. This function MUST use the Square function already written.

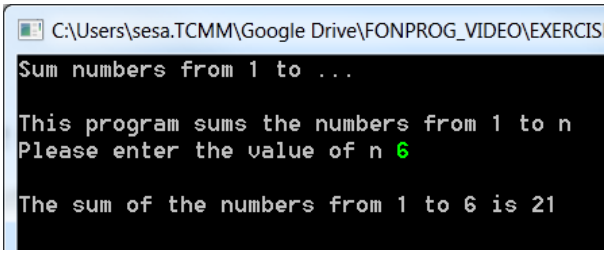
```
public static void Main (string[] args)
{
        int sq;

        Console.WriteLine ("The square of the sums of the numbers in [1, 4]");
        Console.WriteLine ();

        for (int a = 1; a <= 4; a++) {
                for (int b = 1; b <= 4; b++) {
                   sq = SquareOfTheSum (a, b);
                   Console.WriteLine ("{0} + {1} squared equals {2}", a, b, sq);
                }
        }

        Console.SetCursorPosition (0, Console.WindowHeight-1);
        Console.Write ("Press any key to exit");
        Console.ReadKey ();
}

static int Square (int num) {
        int result;

        result = num * num;

        return result;
}

// write here the function SquareOfTheSum that given two integer numbers
// computes the square of their sum.
// This function MUST use the Square function
```



6. Write the code of the function SumUpTo that computes the sum 1+2+3+...+last where last is its parameter. Notice that main already contains an invocation of this function

```
public static void Main (string[] args)
{
        int n;
        int sum;

        Console.WriteLine ("Sum numbers from 1 to ...");
        Console.WriteLine ();

        Console.WriteLine ("This program sums the numbers from 1 to n");
        Console.Write ("Please enter the value of n ");
        Console.ForegroundColor = ConsoleColor.Green;
        n = Convert.ToInt32(Console.ReadLine ());
        Console.ResetColor ();

        sum = SumUpTo (n);

        Console.WriteLine ();
        Console.WriteLine ("The sum of the numbers from 1 to {0} is {1}", n,
sum);

        Console.SetCursorPosition (0, Console.WindowHeight-1);
        Console.Write ("Press any key to exit");
        Console.ReadKey ();
}

// write here the code of the function SumUpTo that
// computes the sum 1+2+3+...+last where last is its parameter
```





7. Complete the program that given a number determines if it is a perfect square or not. A boolean function PerfectSquare is provided (you don't have to understand it; you just have to use it)

```
public static void Main (string[] args)
{
        int n;
        // declare more variables if needed

        Console.WriteLine ("Determine if a number is a perfect square");
        Console.WriteLine ();

        Console.Write ("Please enter the value of n (positive) ");
        Console.ForegroundColor = ConsoleColor.Green;
        n = Convert.ToInt32(Console.ReadLine ());
        Console.ResetColor ();

        Console.WriteLine ();
        Console.Write ("Is the number {0} a perfect square? ", n);

        /*      Write here the piece of code needed to write
                YES, IT'S A PERFECT SQUARE
                or
                NO, IT'S NOT A PERFECT SQUARE
                depending on whether n is a perfect square or not.
                You must use the function PerfectSquare
        */


        Console.SetCursorPosition (0, Console.WindowHeight-1);
        Console.Write ("Press any key to exit");
        Console.ReadKey ();

}

static bool PerfectSquare (int number) {
        // this function determines whether its parameter is a perfect square or not

        int root;

        root = 1;
        while (root * root < number) {
                root++;
        }
        // when here root*root >= number
        // if root*root == number then number is a perfect square otherwise it is not
        if (root * root == number) {
                return true;
        } else {
                return false;
        }
}
```

```
C:\Users\sesa.TCMM\Google Drive\FONPROG_VIDEO\EXERCISES\04_PROCEDURES\SesLab\Multiple...
Determine if a number is a perfect square

Please enter the value of n (positive) 26

Is the number 26 a perfect square? NO, IT'S NOT A PERFECT SQUARE
```

```
C:\Users\sesa.TCMM\Google Drive\FONPROG_VIDEO\EXERCISES\04_PROCEDURES\SesLab\Multiple...
Determine if a number is a perfect square

Please enter the value of n (positive) 25

Is the number 25 a perfect square? YES, IT'S A PERFECT SQUARE
```

8. First, write a function that determines if two integers are relatively prime or not (they are relatively prime if their gcd is 1). The gcd function is already written.
Then, complete the program that tells whether two given numbers are relatively prime or not.

```
public static void Main (string[] args)
{
        int firstNumber, secondNumber;
        // add more variables if needed

        Console.WriteLine ("Determine if two numbers are relatively prime or not");
        Console.WriteLine ();

        Console.Write ("Please enter the first number (positive) ");
        Console.ForegroundColor = ConsoleColor.Green;
        firstNumber = Convert.ToInt32(Console.ReadLine ());
        Console.ResetColor ();
        Console.Write ("Now enter the second number (positive) ");
        Console.ForegroundColor = ConsoleColor.Green;
        secondNumber = Convert.ToInt32(Console.ReadLine ());
        Console.ResetColor ();

        Console.WriteLine ();
        Console.WriteLine ("The numbers {0} and {1} ", firstNumber, secondNumber);

        /* PART TWO
                Write here the code necessary to inform the user of whether
                firstNumber and secondNumber are relatively primer or not.
                You must use a function developed by you */



        Console.SetCursorPosition (0, Console.WindowHeight-1);
        Console.Write ("Press any key to exit");
        Console.ReadKey ();
}

static int Gcd (int a, int b) {
        // computes tge gcd (greatest common divisor) of a and b
        // using Euclid's algorithm
        // Euclid's algorithm: substract until they're equal. That's the gcd

        while (a != b) {
                if (a > b) {
                        a = a - b;
                } else {
                        b = b - a;
                }
        }
        // when here a==b

        return a;
}

/* PART ONE
        write here the function that determines if two integer numbers
        are relatively prime or not. Two numbers are relatively prime
        if their gcd is 1.
        you must use the gcd function
*/
```

```
C:\Users\sesa.TCMM\Google Drive\FONPROG_VIDEO\EXERCISES\04_PROCED
Determine if two numbers are relatively prime or not

Please enter the first number (positive) 25
Now enter the second number (positive) 36

The numbers 25 and 36
ARE RELATIVELY PRIME
```

```
C:\Users\sesa.TCMM\Google Drive\FONPROG_VIDEO\EXERCISES\04_PROCED
Determine if two numbers are relatively prime or not

Please enter the first number (positive) 45
Now enter the second number (positive) 81

The numbers 45 and 81
ARE NOT RELATIVELY PRIME
  Their gcd is 9
```

Exercise 4A


```
public static void MyName ()
{
    // void => return no value
    // () => no parameters (no information from the outside required

    Console.ForegroundColor = ConsoleColor.Yellow;
    Console.WriteLine("Enric Sesa i Nogueras");
    Console.ResetColor();
}
```

Exercise 4B

```
public static void MyNameWithSpaces (int spaces)
{
    // void => return no value
    // spaces => parameter (information coming from the outside)

    // first write the required spaces...
    for (int b=1; b<=spaces; b++)
    {
        Console.Write(" ");
    }

    //... then write the name in yellow
    Console.ForegroundColor = ConsoleColor.Yellow;
    Console.WriteLine("Enric Sesa i Nogueras");
    Console.ResetColor();
}
```

THE VALUE PASSED TO THE PROCEDURE

Alternative solution...

```
public static void MyName()
{
    // void => return no value
    // () => no parameters (no information from the outside required

    Console.ForegroundColor = ConsoleColor.Yellow;
    Console.WriteLine("Enric Sesa i Nogueras");
    Console.ResetColor();
}

public static void MyNameWithSpaces (int spaces)
{
    // void => return no value
    // spaces => parameter (information coming from the outside)

    // first write the required spaces...
    for (int b=1; b<=spaces; b++)
    {
        Console.Write(" ");
    }

    MyName();
}
```

A PROCEDURE MAY USE OTHER PROCEDURES

Exercise 5A

```
public static void Main(string[] args)
{
    // declare variables if needed...
    int numberSquared;

    Console.WriteLine("The squares of the numbers in [0, 10]");
    Console.WriteLine();

    // write  here the code necessary for writing the squares of the numbers in [0, 10]
    // you must use the Square function

    for (int number = 0; number<=10; number++)
    {
        numberSquared = Square(number); // invoke square with parameter number
                                        // and store result returned in numberSquared
        Console.WriteLine("{0} squared equals {1}",number, numberSquared);
    }

    Console.SetCursorPosition(0, Console.WindowHeight - 1);
    Console.Write("Press any key to exit");
    Console.ReadKey();
}
```

Exercise 5B (three different possibilities)

```
static int SquareOfTheSum (int a, int b)
{
    // int => the result returned is an integer
    // a, b => the information required (parameters)

    int sum, result; // local variables

    sum = a + b;
    result = Square(sum); // invoke the Square function

    return result;
}
```

Or...

```
static int SquareOfTheSum (int a, int b)
{
    int sum; // local variable

    sum = a + b;
    return Square(sum);
}
```

Or...

```
static int SquareOfTheSum (int a, int b)
{
    return Square(a + b);
}
```

Exercise 6

```
public static int SumUpTo(int last)
{
    int sum;

    sum = 0;
    // unfold interval [1, number]
    for (int number = 1; number <= last; number++)
    {
        sum = sum + number;
    }

    return sum;
}
```

Exercise 7

```
Console.Write("Please enter the value of n (positive) ");
Console.ForegroundColor = ConsoleColor.Green;
n = Convert.ToInt32(Console.ReadLine());
Console.ResetColor();

Console.WriteLine();
Console.Write("Is the number {0} a perfect square? ", n);

if (PerfectSquare(n))
{
    Console.WriteLine("YES, IT'S A PERFECT SQUARE");
}
else
{
    Console.WriteLine("NO, IT'S NOT A PERFECT SQUARE");
}
```

BOOL FUNCTIONS CAN BE USED IN IF CONDITIONS

ALSO

```
if (PerfectSquare(n) == true)
```

Exercise 8. Part One (three different possibilities)

```
static bool RelativelyPrime (int numOne, int numTwo)
{
    int theGcd;

    theGcd = Gcd(numOne, numTwo);

    // if the gcd is one, the two numbers are relatively prime
    if (theGcd==1) { return true; }
    else { return false; }
}
```

Or...

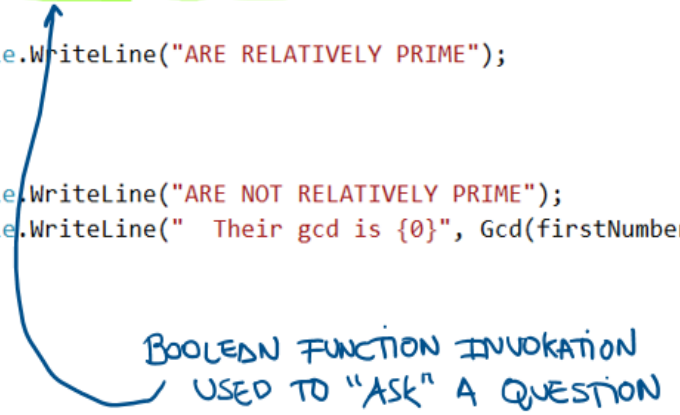
```
static bool RelativelyPrime (int numOne, int numTwo)
{

    // if the gcd is one, the two numbers are relatively prime
    if (Gcd(numOne, numTwo)==1) { return true; }
    else { return false; }
}
```

Or...

```
static bool RelativelyPrime (int numOne, int numTwo)
{
    return Gcd(numOne, numTwo) == 1;
}
```

Exercise 8. Part Two

```
if (RelativelyPrime(firstNumber, secondNumber))
{
    Console.WriteLine("ARE RELATIVELY PRIME");
}
else
{
    Console.WriteLine("ARE NOT RELATIVELY PRIME");
    Console.WriteLine("  Their gcd is {0}", Gcd(firstNumber, secondNumber));
}
```



"Writing" exercises. OTHER

Solutions for some of this exercises are given at the end of the section

9. Write a function that given two integer numbers returns the maximum. Do not use Math.max (This is like writing your own Math.max function). Your function should have this header:

```
static int MyMax(int a, int b)
```

The execution entry point given (main) will help you test your function

10. Write a function that given three integer numbers returns the maximum. Name it MaxOfThree. Use the function of the preceding exercise to write this one. Write a short program to test the behaviour of your function (you can adapt the one given for the previous exercise).

11. Write a function that given an integer number "tells" whether it is even or not. Name it EvenNumber.

12. Write a program that gets four numbers from its user and then prints the sum of the evens and the sum of the odds. Use the function developed in the preceding exercise.

```
Please enter an integer number 12
Please enter an integer number 13
Please enter an integer number 15
Please enter an integer number 21

Sum of even numbers: 12
Sum of odd numbers: 49
```

13. Write a function that given two integer numbers tells whether they have the same parity or not. Two numbers have the same parity if they are both even or both odd. Name it SameParity. Write a short program to test the behaviour of your function

```
Please enter an integer number 7
Please enter another integer number 8
Please enter yet another integer number 9

The numbers 7 and 8 DO NOT HAVE THE SAME PARITY

The numbers 8 and 9 DO NOT HAVE THE SAME PARITY

The numbers 7 and 9 HAVE THE SAME PARITY
```

14. Write a function that given a double number computes its absolute value

15. Write a program that gets two numbers from its user and then prints the absolute value of the sum of the numbers and the sum of their absolute values. Use the function developed in the preceding exercise.

```
Plase enter first number: 12
Plase enter second number: -23

The sum of absolute values (|12| + |-23|)  is 35
The absolute value of the sum (|12 + -23|) is 11
```

16. Write a function that given two integer numbers (>=0) a and b, computes a^b. Do not use Math.pow (it does not provide an integer result). Just iterate. Write a short program to test your function.

17. Write a function that calculates the factorial of a positive (>=0) number (remember that 0!= 1)

18. Write a program that given an integer number n (n>=0) writes the table of the factorials in the interval [0, n]. Base your program in the previous function.

```
FACTORIAL TABLE
--------- -----

Up to which number do you wish the table produced? 9

Factorial(0) = 1
Factorial(1) = 1
Factorial(2) = 2
Factorial(3) = 6
Factorial(4) = 24
Factorial(5) = 120
Factorial(6) = 720
Factorial(7) = 5040
Factorial(8) = 40320
Factorial(9) = 362880
```

19. Write a function that given an integer number n (n>0) determines whether it is prime or not. Hint: to determine if a number is prime search divisors in the interval [2,n/2]. If none is found then the number is prime otherwise it is not. 1 and 2 may be special cases.

20. Based on the previous function, write a program that prints the list of the p first prime numbers. Value will be given by the user.

```
PRIME NUMBERS
----- -------

How many prime numbers do you wish to see? 10

These are the first 10 prime numbers:

2, 3, 4, 5, 7, 11, 13, 17, 19 and 23.
```

```
PRIME NUMBERS
----- -------

How many prime numbers do you wish to see? 100

These are the first 100 prime numbers:

2, 3, 4, 5, 7, 11, 13, 17, 19, 23,
29, 31, 37, 41, 43, 47, 53, 59, 61, 67,
71, 73, 79, 83, 89, 97, 101, 103, 107, 109,
113, 127, 131, 137, 139, 149, 151, 157, 163, 167,
173, 179, 181, 191, 193, 197, 199, 211, 223, 227,
229, 233, 239, 241, 251, 257, 263, 269, 271, 277,
281, 283, 293, 307, 311, 313, 317, 331, 337, 347,
349, 353, 359, 367, 373, 379, 383, 389, 397, 401,
409, 419, 421, 431, 433, 439, 443, 449, 457, 461,
463, 467, 479, 487, 491, 499, 503, 509, 521 and 523.
```

Exercise 9. Two possible solutions

```
static int MyMax(int a, int b)
{
    int maximum;

    if (a>b)
    {
        maximum = a;
    }
    else
    {
        maximum = b;
    }

    return maximum;
}
```

```
static int MyMax(int a, int b)
{
    if (a>b)
    {
        return a;
    }
    else
    {
        return b;
    }
}
```

Exercise 10. Several possible solutions

```
static int MaxOfThree (int a, int b, int c) {
    int maximum;

    maximum = MyMax (b, c);

    maximum = MyMax (a, maximum);

    return maximum;
}
```

```
static int MaxOfThree (int a, int b, int c) {
    int inter;

    inter = MyMax (b, c);

    return MyMax (a, inter);
}
```

```
static int MaxOfThree (int a, int b, int c) {
    return MyMax (a, MyMax (b, c));
}
```

Exercise 11. Several possible solutions

```
static bool EvenNumber (int number) {
    bool answer;

    if (number % 2 == 0) {
        answer = true;
    } else {
        answer = false;
    }

    return answer;
}
```

```
static bool EvenNumber (int number) {
    if (number % 2 == 0) {
        return true;
    } else {
        return false;
    }
}
```

```
static bool EvenNumber (int number) {
    return number % 2 == 0;
}
```

Exercise 12

```
public static void Main (String [] args) {
    int theNumber;
    int sumEven, sumOdd;

    sumEven = 0;
    sumOdd = 0;

    for (int i = 1; i <= 4; i++) {
        Console.Write ("Please enter an integer number ");
        theNumber = Convert.ToInt32 (Console.ReadLine());
        if (EvenNumber (theNumber)) {
            sumEven = sumEven + theNumber;
        } else {
            sumOdd = sumOdd + theNumber;
        }
    }
    // when here all four numbers have been entered and processed
    Console.WriteLine ();
    Console.WriteLine ("Sum of even numbers: {0}", sumEven);
    Console.Write ("Sum of odd numbers: {0}", sumOdd);
```

Exercise 13. Several possible solutions

```
static bool SameParity(int n1, int n2) {
    int rem1, rem2;

    rem1 = n1 % 2;
    rem2 = n2 % 2;

    if (rem1==rem2) { return true; }
    else { return false; }
}
```

```
static bool SameParity(int n1, int n2) {
    int rem1, rem2;

    rem1 = n1 % 2;
    rem2 = n2 % 2;

    return rem1 == rem2;
}
```

```
static bool SameParity(int n1, int n2)
{
    return n1 % 2 == n2 % 2;
}
```

Exercise 14. Two different possibilities

```
static double AbsoluteValue (double n) {
    double absVal;

    if (n < 0) {
        absVal = -n;
    } else {
        absVal = n;
    }

    return absVal;
}
```

```
static double AbsoluteValue (double n) {
    if (n < 0) {
        return -1 * n;
    } else {
        return n;
    }
}
```

Exercise 15

```
double num1, num2;
double sumOfAbsVal, absValOfSum;

Console.Write ("Plase enter first number: ");
num1 = Convert.ToDouble (Console.ReadLine());
Console.Write ("Plase enter second number: ");
num2 = Convert.ToDouble (Console.ReadLine());
Console.WriteLine ();

sumOfAbsVal = AbsoluteValue (num1) + AbsoluteValue (num2);

absValOfSum = AbsoluteValue (num1 + num2);

Console.WriteLine ("The sum of absolute values (|{0}|+|{1}|) is {2}",num1, num2,sumOfAbsVal);
Console.Write ("The absolute value of the sum (|{0} + {1}|) is {2}", num1, num2, absValOfSum);

Console.SetCursorPosition (0, Console.WindowHeight - 1);
Console.Write ("Press any key to exit");
Console.ReadKey (true);
```

Exercise 16. Function and detail of main

```
static int Power (int a, int b) {
    // a base
    // b exponent

    int result;

    result = 1;

    // multiply by a b times
    for (int i = 1; i <= b; i++) {
        result = result * a;
    }

    return result;
}
```

```
public static void Main (string [] args) {

    Console.WriteLine ("THE POWERS OF TWO");
    Console.WriteLine ("-----------------");
    Console.WriteLine ();

    for (int exponent = 0; exponent <= 10; exponent++) {
        Console.WriteLine (" 2 exponent {0} is {1} ", exponent, Power(2, exponent));
    }
```

Exercises “split” into multiple parts

No solution is provided for the exercises in this section

21. Write a program that is able to draw the perimeter of squares and rectangles. The user decides the shape and the dimensions.
Decompose the problem into the following subproblems:

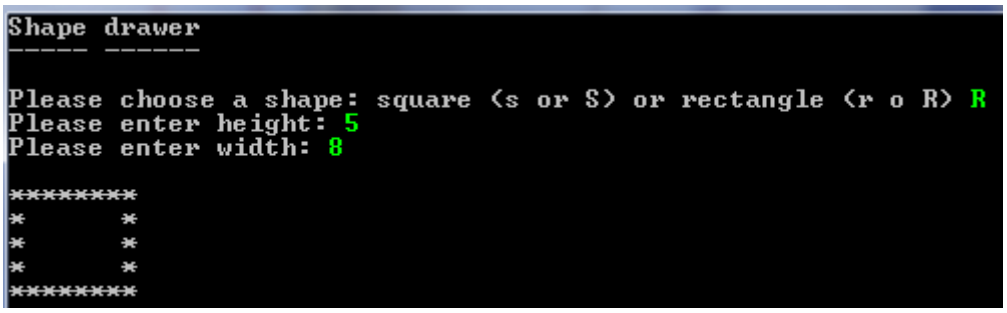
(a) Write a procedure that given an int number (t) and a character (c) writes t times the character c (in a single line, without using cursor positioning). This procedure should have the following header:

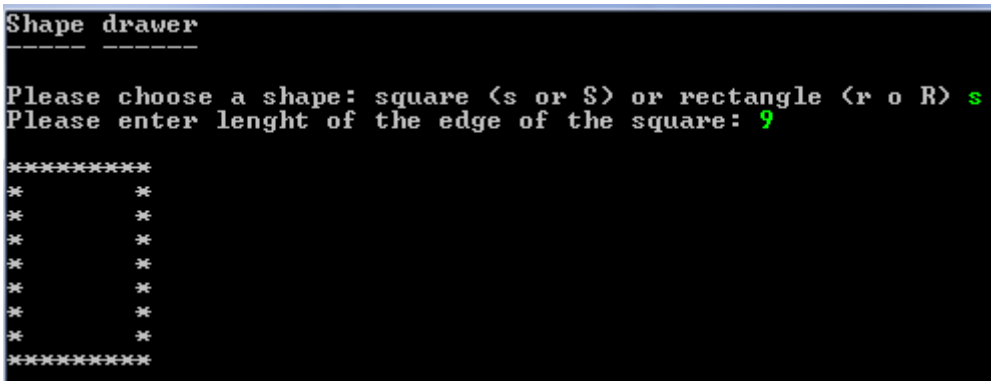
```
static void WriteTimes (int t, char c)
```

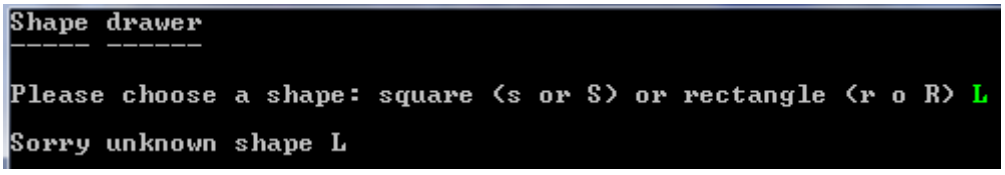
(b) Write a procedure that given an int number n writes the perimeter of a nxn square. Use the procedure developed in the previous section.

(c) Write a procedure that given two integer parameters w and h, writes the perimeter of a wxh rectangle (w: width; h:height).

(d) Write the program using the procedures developed in the previous sections





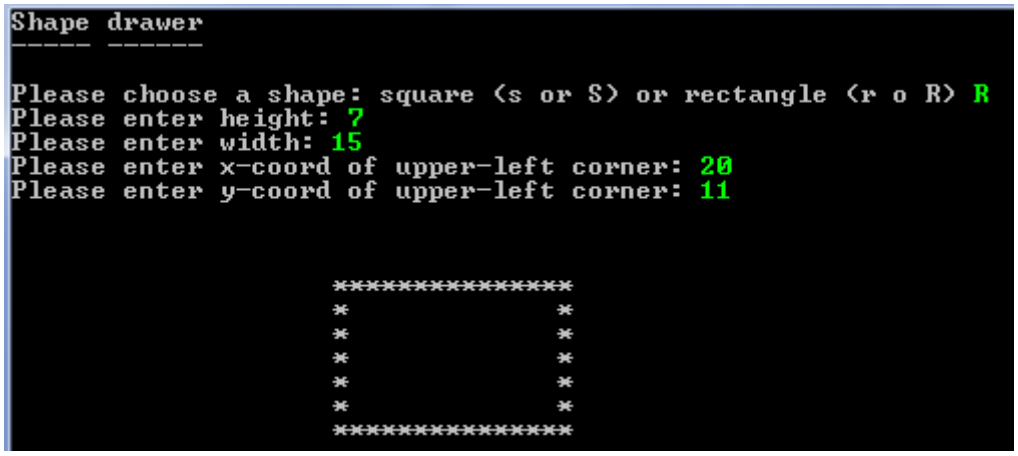


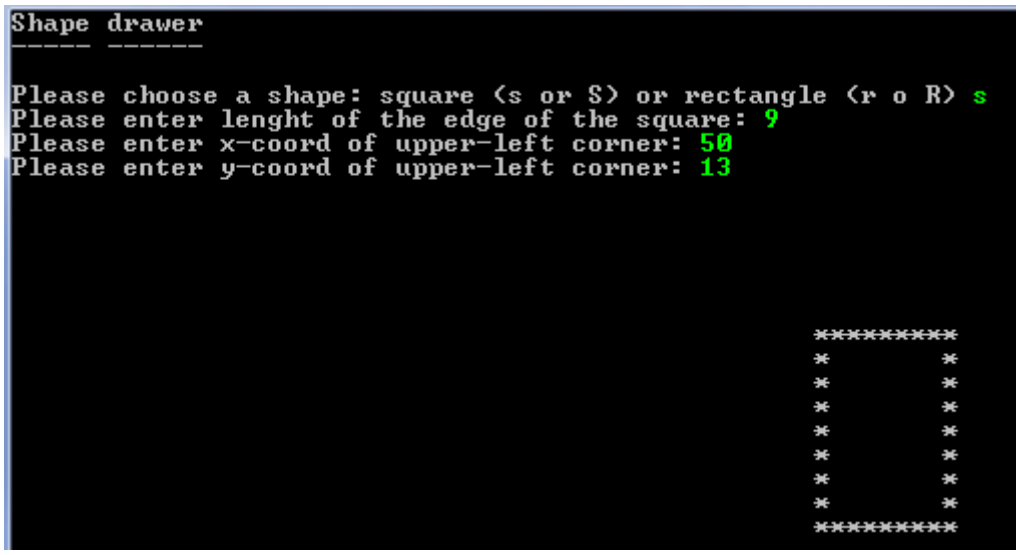
22. Same as before but now the user not only specifies the shape (square or rectangle) and its dimensions but also the coordinates (x,y) of the upper-left corner.

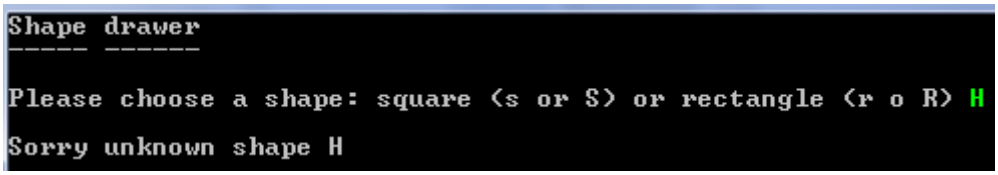
(a) Change the procedure that writes a given character a given number of times. Now, it has to have two more parameters (x and y) to specify the initial coordinates. Use cursor positioning. Now the header should be

```
static void WriteTimesWithPosition (int t, char c, int x, int y)
```

(b) Rewrite the procedures for writing the perimeter of a rectangle and the perimeter of a square so that now they can take two more parameters (x and y) to specify the location of the upper-left corner.







23. Write a program that given a positive integer (>=0) computes and shows the sum of its digits. Base your program in function capable of computing the sum of the digits of its parameter. The header of the function must be:

```
static int SumOfDigits (int number)
```

In order to solve this problem, decompose it into the following parts:

(a) write a function that given an integer number computes how many digits it has

```
static int NumberOfDigits (int number)
```

(b) write a function that given an integer exponent computes $10^{exponent}$. Since you want an integer result, do not use Math.pow. Just iterate...

```
static int PowerOfTen (int exponent)
```

(c) write a function with header

```
static int NthDigit (int number, int n)
```

that produces the n^{th} digit (n is the second parameter) of the given number (the first parameter). Consider that digits are numbered from 0 onwards (the digit of the units is the 0^{th} digit, the digits of the tens is the 1^{st}, .. the digit of the thousands is the 3^{rd},...). Hint: the n^{th} digit of number is the last digit of number/10^n (e.g. the 4^{th} digit of 123456 is the last digit of 123456/10^4)

(d) Write sumOfDigits using numberOfDigits and nthDigit to iterate over the digits of the given number

```
Please enter an integer number (>= 0): 0

The sum of the digits of 0 is 0
```

```
Please enter an integer number (>= 0): 7

The sum of the digits of 7 is 7
```

```
Please enter an integer number (>= 0): 12345

The sum of the digits of 12345 is 15
```

```
Please enter an integer number (>= 0): 3303

The sum of the digits of 3303 is 9
```

24. A pair of numbers, p and q, are said to be amicable if the sum of the proper divisors of p is equal to q and the sum of the proper divisors of q is equal to p. A proper divisor of a number is a positive factor of that number other than the number itself (all the divisors of the number except the number itself). For example, the numbers 220 and 284 are a pair of amicable numbers since the sum of the proper divisors of 220 is equal to 284 and the sum of the proper divisors of 284 is equal to 220 (the proper divisors of 220 are 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 and 110 and 1+2+4+5+10+ 11+20+22+44+55+110 = 284; the proper divisors of 284 are 1, 2, 4, 71 and 142 and 1+2+4+71+ 142 = 220).
Write a program that given two positive integer numbers determines whether they are an amicable pair or not.

```
Please enter first number (p>0): 220
Please enter second number (q>0): 284
The numbers 220 and 284 ARE AN AMICABLE PAIR
```

```
Please enter first number (p>0): -589
   Incorrect number. Must be > 0. Please retry -789
   Incorrect number. Must be > 0. Please retry 789
Please enter second number (q>0): 0
   Incorrect number. Must be > 0. Please retry 400
The numbers 789 and 400 ARE NOT AN AMICABLE PAIR
```

```
Please enter first number (p>0): 1184
Please enter second number (q>0): 1210
The numbers 1184 and 1210 ARE AN AMICABLE PAIR
```

```
Please enter first number (p>0): 5020
Please enter second number (q>0): 5564
The numbers 5020 and 5564 ARE AN AMICABLE PAIR
```

```
Please enter first number (p>0): 789
Please enter second number (q>0): 551
The numbers 789 and 551 ARE NOT AN AMICABLE PAIR
```

```
Please enter first number (p>0): 9363584
Please enter second number (q>0): 9437056
The numbers 9363584 and 9437056 ARE AN AMICABLE PAIR
```

In order to solve this problem, decompose it into the following parts:

(a) write a function that given two integers determine if the first is divisible by the second or not (determine if the second is a divisor of the first)

```
static bool Divisible (int num, int candidateDivisor)
```

(b) write a function that given an integer computes the sum of its proper divisors

```
static int SumOfProperDivisors (int num)
```

In the implementation of this function make use of the previous one

(c) write a function that given two positive integers determines if they are an amicable pair of not

```
static bool AmicablePair (int n1, int n2)
```

(d) Write a function with header

```
static int ReadPositiveInteger (String m1, String m2)
```

that gets from the console a positive (>0) integer. While the number entered by the user is not a positive integer, the function insists. Once the number is correct, the function returns it. The first parameter (m1) is the string used to ask for the number the first time. The second one (m2) is the string used to insist.

25. A natural number is said to be perfect if it is equal to the sum of its proper divisors (the sum of its positive divisors excluding itself.

Use the

`static int SumOfProperDivisors (int num)`

function developed in the previous exercise to write a function of header

`static bool IsPerfect (int n)` That determines whether its parameter is a perfect number or not. Then, write a program that given a positive integer checks whether it is perfect or not

```
Please enter a positive ( >0 ) number: 496
The number 496 IS PERFECT
```

```
Please enter a positive ( >0 ) number: -178
   Incorrect number. Must be > 0. Please retry 178
The number 178 IS NOT PERFECT
```

```
Please enter a positive ( >0 ) number: 125
The number 125 IS NOT PERFECT
```

```
Please enter a positive ( >0 ) number: 33550336
The number 33550336 IS PERFECT
```

26. Write a program that given a natural number of exactly 5 digits determines whether it is a palindrome or not. A number is said to be palindrome if it has the same value when read forwards or backwards. For instance 12321 and 79597 are five-digits palindromes while 12312 and 81217 are not.

In order to solve this problem, decompose it into the following parts (some of this subproblems may have been solved in previous exercises):

(a) write a function with header

```
static int NthDigit (int number, int n)
```

that produces the n^{th} digit (n is the second parameter) of the given number (the first parameter). Consider that digits are numbered from 0 onwards with the digits of the units being the 0^{th}

(b) write a function that given a five-digit number reverses it.

```
static int Reverse (int number)
```

(to reverse a five-digit number, units have to become tens of thousand, tens have to become thousands, hundreds do not change their role, thousands have to become tens and tens of thousand have to become units. That is:
Reversed number = units·10000 + tens·1000 + hundreds·100 + thousands ·10 + tens of thousand.

(c) Write a function that given a five-digit number checks whether it is palindrome or not.

(d) Write a function that gets from the console an integer in the interval [min, max] (with min and max being the parameters of this function). While the number entered by the user is outside the interval, the function insists. Once the number is correct, the function returns it.

```
Please enter an integer in [10000, 99999] 12789
The number 12789 IS NOT A PALINDROME
```

```
Please enter an integer in [10000, 99999] 18781
The number 18781 IS A PALINDROME
```

```
Please enter an integer in [10000, 99999] 5005
   Given number is not in [10000, 99999]. Please reenter it 50005
The number 50005 IS A PALINDROME
```

Parameters by reference “Reading” exercises

Solutions provided at the end of the section

27. Consider the following procedure

```
static void Equalize (int model, bool add, ref int a, ref int b)  {
    model = model % 2;

    if ((a+b) % 2 != model) {
        if (add) {
            a++;
        } else {
            b--;
        }
    }
}
```

What will the following fragments of programs write?

(a) fragment 1

```
    int n1, n2;

    n1 = 15;
    n2 = 10;
    Equalize (12, true, ref n1, ref n2);

    Console.WriteLine ("n1 = {0}, n2 = {1}", n1, n2);
```

(b) fragment 2

```
    int n1, n2;

    n1 = 15;
    n2 = 10;
    Equalize (n2, false, ref n1, ref n2);

    Console.WriteLine ("n1 = {0}, n2 = {1}", n1, n2);
```

(c) fragment 3

```
    int a, b;

    a = 5;
    b = 3;
    Equalize (b, false, ref a, ref a);

    Console.WriteLine ("a = {0}, b = {1}", a, b);
```

28. Consider the following procedure

```
static void MultipleOps (ref int a, ref int b, out int c) {
    a++;
    b--;
    c = a + b;
}
```

What will the following fragments of programs write?

(a) fragment 1

```
int a, b, c;

a = 2;
b = 9;

MultipleOps (ref a, ref b, out c);

Console.WriteLine ("a = {0}, b = {1}, c = {2}", a, b, c);
```

(b) fragment 2

```
int b, c;

b = 3;
c = b;

MultipleOps (ref c, ref b, out c);

Console.WriteLine ("b = {0}, c = {1}", b, c);
```

(c) fragment 3

```
int a;

a = 10;

MultipleOps (ref a, ref a, out a);

Console.WriteLine ("a = {0}", a);

Console.ReadKey ();
```

(d) fragment 4

```
int a, b, c;

a = 1;
b = 1;
c = 0;

MultipleOps (ref a, ref b, out c);
MultipleOps (ref c, ref b, out a);
MultipleOps (ref a, ref c, out b);

Console.WriteLine ("a = {0}, b = {1}, c = {2}", a, b, c);
```

Solutions

27.

(a)

n1 = 16, n2 = 10

(b)

n1 = 15, n2 = 9

(c)

a = 4, b = 3

28.

(a)

a = 3, b = 8, c = 11

(b)

b = 2, c = 6

(c)

a = 20

(d)

a = 3, b = 5, c = 2

OTHER EXERCISES

Exercise "Yet another mark calculator"

In a University Course there are four practical assignments (p1, p2, p3, p4), two written exams (w1 and w2) and a practical exam (pe).

The practical mark (PM) is calculated as follows:

$$PM = \begin{cases} (min(p1, p2, p3, p4) + max(p1, p2, p3, p4)) \,/\, 2 & if all\ p_i\ are \geq 4 \\ min(p1, p2, p3, p4) & otherwise \end{cases}$$

If (all practical assignments have marks equal to or higher than 4 then PA is the average of the minimum and the maximum –the minimum plus the maximum divided into 2). If any mark is lower than 4 then PA is the minimum among p1, p2, p3 and P4

The exam mark (EM) is calculated as follows

$$EM = \begin{cases} \dfrac{w1 + w2 + pe}{3} & if\ pe \geq 4 \\ \dfrac{w1 + w2 + pe + pe}{4} & if\ pe < 4\ but\ pe \geq 3 \\ pe & otherwise \end{cases}$$

When the practical exam has a mark equal to or higher than 4 then EM is the average of w1, w2 and Pe. When the practical exam has a mark lower than 4 but not lower than 3 then EM is the average of w1, w2 , pe and pe (pe is added twice). In any other case the value of EM is that of pe.

And, finally, the final mark (FM) is calculated as follows:

FM = 0.4·PM + 0.6·EM

a) Write a function that given the marks of the practical assignments computes the value PM
b) Write a function that given the marks of the three exams computes the value of EM
c) Write an action that given the values of PM and EM
   - Show the values of PM and EM
   - Shows the value of FM
   - Congratulates the student if FM ≥ 9
   - Advices the student to work harder if FM<5
d) Complete the program…

For your convenience, two functions are already provided. You must use them (correctly, of course).

The following images depict the expected behaviour of the program.

```
Marks Calculator
----- ----------

Please enter the value of practical assignment 1 (p1): 5
Please enter the value of practical assignment 2 (p2): 6
Please enter the value of practical assignment 3 (p3): 7
Please enter the value of practical assignment 4 (p4): 8

Please enter the value of first written exam (w1): 5
Please enter the value of second written exam (w2): 6
Please enter the value of prectical exam (pe): 7


Your practical mark (PM) is 6,5 points
Your exam mark (EM) is 6 points
Your final mark (FM) is 6,2 points
```

```
Marks Calculator
----- ----------

Please enter the value of practical assignment 1 (p1): 10
Please enter the value of practical assignment 2 (p2): 10
Please enter the value of practical assignment 3 (p3): 10
Please enter the value of practical assignment 4 (p4): 1

Please enter the value of first written exam (w1): 6
Please enter the value of second written exam (w2): 6
Please enter the value of prectical exam (pe): 3,9


Your practical mark (PM) is 1 points
Your exam mark (EM) is 4,95 points
Your final mark (FM) is 3,37 points

What a pity!!! You should heve worked harder!!!
```

```
Marks Calculator
----- ----------

Please enter the value of practical assignment 1 (p1): 9
Please enter the value of practical assignment 2 (p2): 10
Please enter the value of practical assignment 3 (p3): 9
Please enter the value of practical assignment 4 (p4): 10

Please enter the value of first written exam (w1): 9
Please enter the value of second written exam (w2): 9
Please enter the value of prectical exam (pe): 10


Your practical mark (PM) is 9,5 points
Your exam mark (EM) is 9,33333333333333 points
Your final mark (FM) is 9,4 points

CONGRATULATIONS!!! You've done an excellent job
```

Solution at the end of the section

No solution is provided for the next two exercises.

Exercise “The Visit Card Configurator”

Carefully analyze the five procedures accompanying Main. Then complete main in order to make it work as a visit card configurator. Whenever possible use the procedures provided. Let them help you write a “neat and clean” program.

```
public static void Main (string[] args)
{
    String name, surname, town;
    int age;
    // add variables if needed.

    Console.SetWindowSize (80, 20);

    Console.WriteLine ("VISIT CARD CONFIGURATOR");
    Console.WriteLine ("----- ---- ------------\n");

    /* COMPLETE */


    PressAnyKey ("Press any key to exit...", true, ConsoleColor.Green);
}

static void WriteCentered(int yCoord, String message) {
    int xCoord;

    xCoord = (Console.WindowWidth - message.Length) / 2;

    Console.SetCursorPosition (xCoord, yCoord);
    Console.Write (message);
}

static void WriteCenteredCharRepetitions (int yCoord,
                                          int repetitions, char c) {
    int xCoord;

    xCoord = (Console.WindowWidth - repetitions) / 2;

    Console.SetCursorPosition (xCoord, yCoord);

    for (int i = 1; i <= repetitions; i++) {
        Console.Write (c);
    }
}
```

```
static String AskForAString (String message, bool useGreenForAnswer) {
    String result;
    ConsoleColor oldColor;

    // save current color
    oldColor = Console.ForegroundColor;

    Console.Write (message+' ');
    if (useGreenForAnswer) {
        // change color
        Console.ForegroundColor = ConsoleColor.Green;
    }
    result = Console.ReadLine ();
    if (useGreenForAnswer) {
        // restore old color
        Console.ForegroundColor = oldColor;
    }
    return result;
}

static int AskForANumber (String message, bool useGreenForAnswer) {
    int result;
    ConsoleColor oldColor;

    // save current color
    oldColor = Console.ForegroundColor;

    Console.Write (message+' ');
    if (useGreenForAnswer) {
        // change color
        Console.ForegroundColor = ConsoleColor.Green;
    }
    result = Convert.ToInt32(Console.ReadLine ());
    if (useGreenForAnswer) {
        // restore old color
        Console.ForegroundColor = oldColor;
    }
    return result;
}

static void PressAnyKey (String message, bool bottom, ConsoleColor color) {
    ConsoleColor oldColor;

    oldColor = Console.ForegroundColor;
    Console.ForegroundColor = color;

    if (bottom) {
        Console.SetCursorPosition (0, Console.WindowHeight-1);
    }

    Console.Write (message+' ');

    Console.ForegroundColor = oldColor;

    Console.ReadKey (true);
}
```

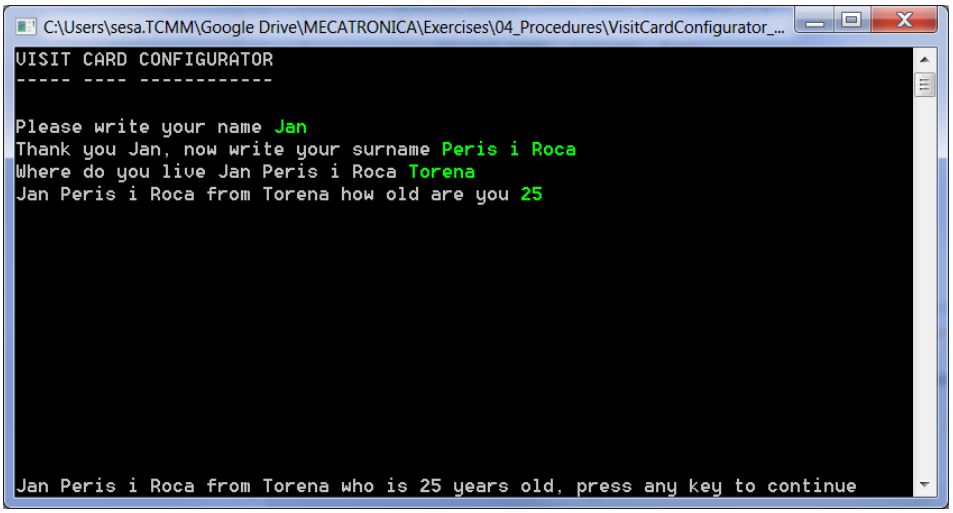


Image 1: first the program asks its user for some basic information...

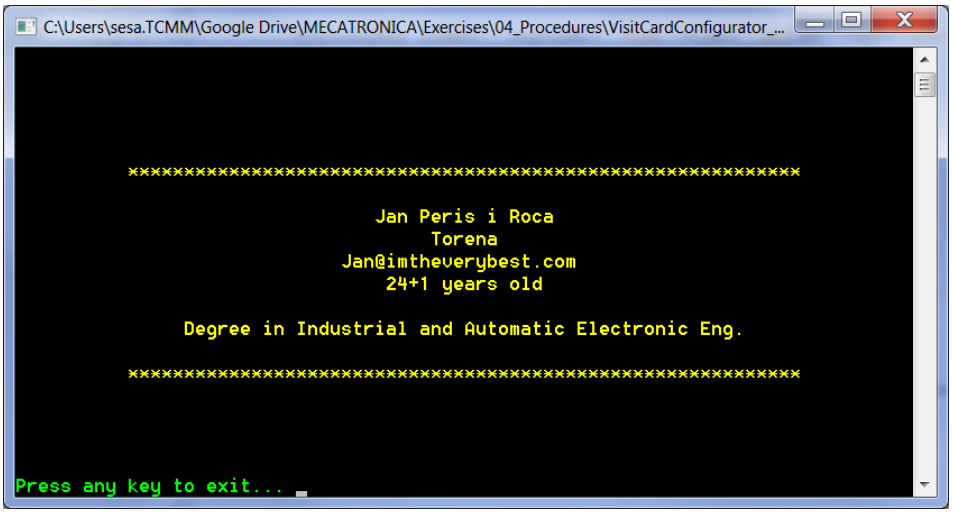


Image 2: ... and then it produces a simple "visit card"

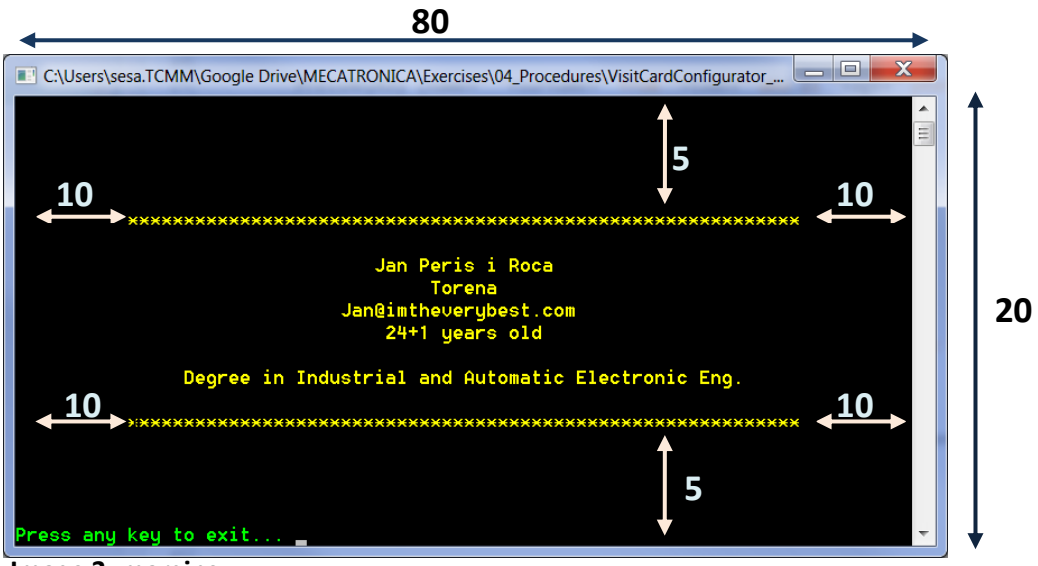


Image 3: margins

Exercise "The Dancing Squares"

This is an exercise in multiple stages.
This is the goal: ***Write a program that draws two squares, one blue, one red, at random positions. The squares are shown for a short time –say a third of a second- and then erased. After a short time –another third of a second- two new squares are drawn at new random positions... The cycle ends when the two squares intersect.*** *Hint: to erase a square redraw it using character whitespace (or using as colour the colour of the background)*

Determining when two squares intersect may seem a complicated problem. It is not if it is decomposed into smaller problems...

1. Write a function with header
   ```
   static bool IsInInterval (int x, int min, int max)
   ```
   that "tells" whether x is in the interval [min, max] or not.

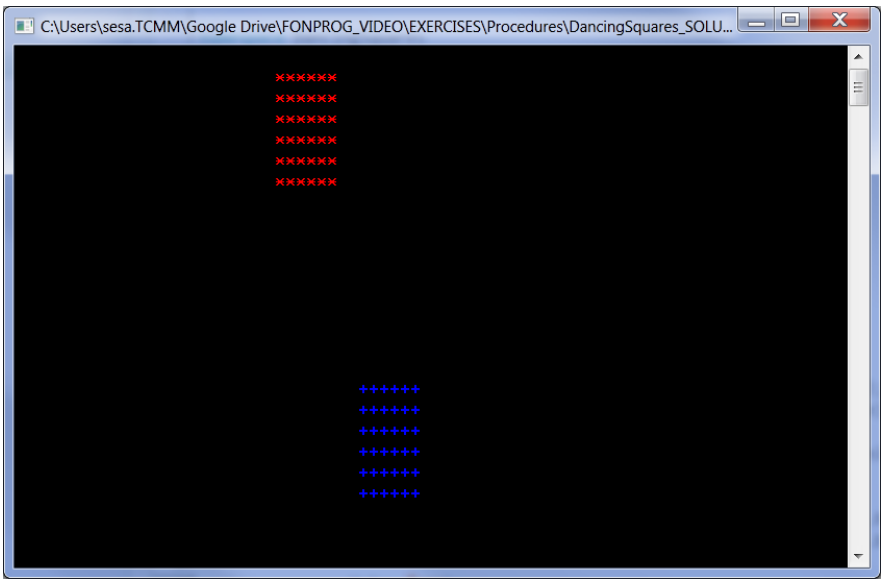
2. Write a function with header
   ```
   static bool IsInside (int sqX, int sqY, int sqSize, int x, int y)
   ```
   sqX and sqY are the coordinates of the upper left corner of a square. The length of any edge of this square is sqSize. x and y are the coordinates of a point. The function returns true if the point (x,y) is inside the square and false otherwise. HINT: point (x,y) is inside the square if and only if x is in the interval [sqX, sqX+sqSize-1] and y is in the interval [sqY, sqY+sqSize-1]. Use the function of the preceding stage

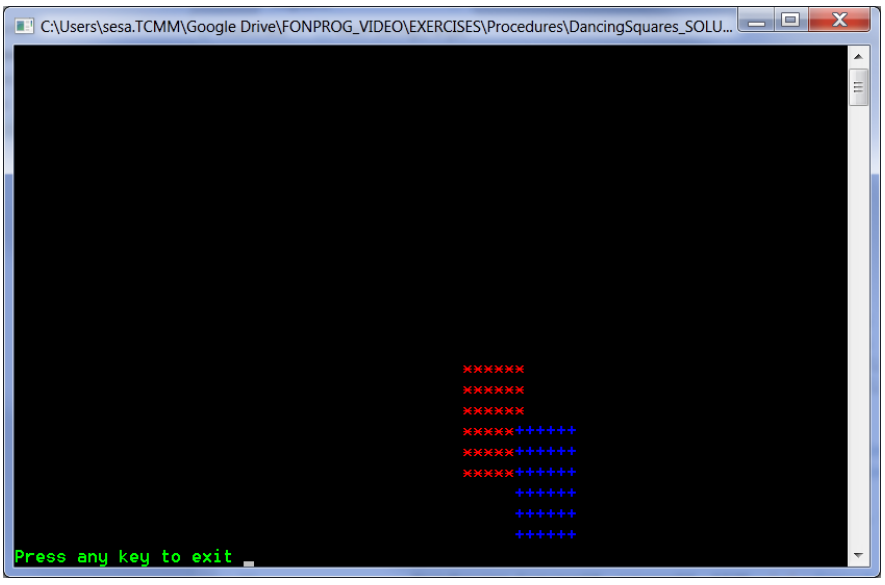
3. Write a function with header
   ```
   static bool Intersect (int sqX1, int sqY1, int sqSize1,
                          int sqX2, int sqY2, int sqSize2)
   ```
   sqX1 and sqY1 are the coordinates of the upper left corner of square the edges of which have a length of sqSize1. sqX2 and sqY2 are the coordinates of the upper left corner of another square the edges of which have a length of sqSize2. The function returns true if the two squares intersect and false otherwise. HINT: the two squares intersect if and only if any of the corners of the second is inside the first (use the function of the preceding stage).

4. Write an action with the following header
   ```
   static void DrawSquare (int x, int y, int le, char c, ConsoleColor color)
   ```
   x and y are the coordinates of the upper left corner of a square of size le. The procedure draws this square, in the colour given, using character c.

5. Write the program


C:\Users\sesa.TCMM\Google Drive\FONPROG_VIDEO\EXERCISES\Procedures\DancingSquares_SOLU...


C:\Users\sesa.TCMM\Google Drive\FONPROG_VIDEO\EXERCISES\Procedures\DancingSquares_SOLU...


C:\Users\sesa.TCMM\Google Drive\FONPROG_VIDEO\EXERCISES\Procedures\DancingSquares_SOLU...
Press any key to exit

// Yet another mark calculator. Solution

```
/* a)   Write a function that given the marks
   of the practical assignments computes the value PM */

static double ComputePracticalMark (double pm1, double pm2, double pm3, double pm4)
{
    double min, max;
    double result;

    if (pm1>=4 && pm2>=4 && pm3>=4 && pm4>=4)
    {
        min = MaxMinOfFour(pm1, pm2, pm3, pm4, false);
        max = MaxMinOfFour(pm1, pm2, pm3, pm4, true);
        result =  (min + max) / 2.0;
    }
    else
    {
        min = MaxMinOfFour(pm1, pm2, pm3, pm4, false);
        result = min;
    }
    return result;
}
/* b)   Write a function that given the marks
 * of the three exams computes the value of EM */

static double ComputeExamMark (double w1, double w2, double pe)
{
    double result;

    if (pe>=4)
    {
        result = (w1 + w2 + pe) / 3;
    }
    else if (pe>=3)
    {
        result = (w1 + w2 + pe + pe) / 4;
    }
    else
    {
        result = pe;
    }

    return result;
}
```

ALL (88) ARE 4 OR MORE

MINIMUM

MAXIMUM

NOT ALL ARE 4 OR MORE (= AT LEAST ONE IS LESS THAN 4)

FOUR OR MORE

THREE OR MORE (BUT LESS THAN FOUR)

LESS THAN THREE

// solution continues on next page

```
/* c)   Write an action that given the values of PM and EM
        -   Show the values of PM and EM
        -   Shows the value of FM
        -   Congratulates the student if FM ≥ 9
        -   Advices the student to work harder if FM<5
*/
static void ShowMarks(double pm, double em)
{
    double fm;

    fm = 0.4 * pm + 0.6 * em;

    Console.Write("Your practical mark (PM) is ");
    Console.ForegroundColor = ConsoleColor.Yellow; Console.Write(pm); Console.ResetColor();
    Console.WriteLine(" points");
    Console.Write("Your exam mark (EM) is ");
    Console.ForegroundColor = ConsoleColor.Yellow; Console.Write(em); Console.ResetColor();
    Console.WriteLine(" points");
    Console.Write("Your final mark (FM) is ");
    Console.ForegroundColor = ConsoleColor.Yellow; Console.Write(fm); Console.ResetColor();
    Console.WriteLine(" points");
    Console.WriteLine();

    if (fm >= 9)
    {
        Console.WriteLine("CONGRATULATIONS!!! You've done an excellent job");
    }
    else if (fm < 5)
    {
        Console.WriteLine("What a pity!!! You should have worked harder!!!");
    }
}
```

NO VALUE RETURNED. THIS IS AN ACTION

No return ANYWHERE

// main

```
public static void Main(string[] args)
{
    double p1, p2, p3, p4;
    double w1, w2, pe;
    double pm, em;

    Console.WriteLine("Marks Calculator");
    Console.WriteLine("----- ----------");
    Console.WriteLine();

    /* COMPLETE */

    p1 = ReadMark("practical assignment 1 (p1)");
    p2 = ReadMark("practical assignment 2 (p2)");
    p3 = ReadMark("practical assignment 3 (p3)");
    p4 = ReadMark("practical assignment 4 (p4)");
    Console.WriteLine();
    w1 = ReadMark("first written exam (w1)");
    w2 = ReadMark("second written exam (w2)");
    pe = ReadMark("practical exam (pe)");
    Console.WriteLine("\n");

    pm = ComputePracticalMark(p1, p2, p3, p4);
    em = ComputeExamMark(w1, w2, pe);

    ShowMarks(pm, em);

    Console.SetCursorPosition(0, Console.WindowHeight - 1);
    Console.Write("Please press any key to exit");
    Console.ReadKey(true);
}
```

ShowMarks needs the practical work and the exam mark calculated beforehand (the functions do the work)

ARRAYS

"Reading" exercises

Solutions at the end of the section

1. Consider the following program

```
public static void Main (string[] args)
{
    int[] t;
    int i, sum;

    t = new int[7];

    i = 0;
    while (i < 7) {
        t [i] = i % 2;
        i++;
    }

    sum = 0;
    for (int j = 0; j < 7; j = j + 2) {
        sum = sum + t [j];
    }

    Console.WriteLine ("sum = {0}", sum);

}
```

What will it write? What will the contents of array t be?

2. Consider the following program

```
public static void Main (String [] args) {

    int[] t = new int[5];
    int sum, idx;

    for (int i = 0; i <= 4; i++) {
        if (i % 2 == 0) {
            t [i] = i;
        } else {
            t [i] = 2 * i;
        }
    }

    sum = 0;
    idx = 0;
    while (!(idx == 4)) {
        sum = sum + t [idx];
        idx++;
    }

    Console.WriteLine ("sum = {0}", sum);

}
```

What will it write? What will the contents of array t be?

3. Consider the following program

```
public static void Main (String [] args) {
  int[] t1, t2;

    t1 = new int[4];
    t2 = new int[10];

    for (int i = 0; i < t2.Length; i++) {
        t2 [i] = 0;
    }

    t1 [0] = 1;
    for (int i = 1; i < t1.Length; i++) {
        t1 [i] = i + t1 [i - 1];
    }

    for (int i = 0; i < t1.Length; i++) {
        t2 [t1 [i]] = 2 * i;
    }

    for (int i = 0; i < t2.Length; i++) {
        Console.WriteLine ("t2[{0}] = {1}",i, t2[i]);
    }

}
```

What will it write? What will the contents of arrays t1 and t2 be?

4. Consider the following (fragment of) program

```
int[] arrayOfInts;
double[] arrayOfDoubles;
int idx;

arrayOfInts = new int[4];
arrayOfDoubles = new double[8];

idx = 0;
while (idx < arrayOfDoubles.Length) {
    arrayOfDoubles [idx] = 10.0 * idx;
    idx = idx + 1;
}

for (idx = 0; idx <= arrayOfInts.Length - 1; idx++) {
    arrayOfInts [idx] = 2 * idx + 1;
}

idx = 1;
while (idx < arrayOfInts.Length - 1) {
    arrayOfInts [idx] = arrayOfInts [idx] - 1;
    idx++;
}

/* POINT 1 */


for (idx = 0; idx < arrayOfInts.Length; idx = idx + 2) {
    arrayOfDoubles [arrayOfInts [idx]]++;
}

for (idx = 1; idx < arrayOfInts.Length; idx = idx + 2) {
    arrayOfDoubles [arrayOfInts [idx]]--;
}

/* POINT 2 */
```

What will the contents of arrayOfDoubles and arrayOfInts be when it reaches

a. POINT 1?

b. POINT 2?

5. Consider the following program

```
public static void Main (string[] args)
{
    int size;
    int[] vector;

    vector = new int[10];

    Console.Write ("Enter size: ");
    size = Convert.ToInt32(Console.ReadLine ());

    Filler (vector, size);

    FirstPoker (vector [size - 1]);
    FirstPoker (vector [size - 2]);

    for (int i = 0; i<size; i++) {
        SecondPoker(vector, i);
    }

    Console.WriteLine ();
    ShowArrayFirstElements (vector, size, true);

    Console.ReadKey (true);
}

static void Filler (int[] v, int m)  {
    for (int nd = 0; nd < m; nd++) {
        v [nd] = nd;
    }
}

static void FirstPoker (int value) {
    value = 2 * value + 1;
}

static void SecondPoker (int[] v, int pos) {
    v [pos] = 3 * v [pos];
}

static void ShowArrayFirstElements (int[] anArray, int numPos, bool vertical)
{ // show the first numPos elements of the array
    if (vertical) {
        for (int i = 0; i < numPos; i++) {
            Console.WriteLine ("[{0}]: \t{1}", i, anArray [i]);
        }
    } else {
        for (int i = 0; i < numPos - 1; i++) {
            Console.Write ("{0}, ", anArray [i]);
        }
        Console.WriteLine ("{0}. ", anArray [numPos - 1]);
    }
}
```

What will it write? What will the contents of array vector be when

a. The value given for size is 4?
b. The value given for size is 12?

6. Consider the following program

```
public static void Main (String [] args) {
    int[] t1, t2, t3;
    int ad1, ad2;

    t1 = new int[10];
    t2 = new int[6];

    Filler (t1, 2);
    Filler (t2, AddUp (t1, 0, 3));

    /* POINT 1 */

    t3 = new int[3];

    Filler (t3, 2);
    Poker (t3, t2, t1);

    ad1 = AddUp (t1, 0, t1.Length - 2);
    ad2 = AddUp (t2, 0, t2.Length - 1);

    /* POINT 2 */

}


static void Filler (int[] t, int v) {
    for (int i = 0; i < t.Length; i++) {
        t [i] = v * i;
    }
}

static int AddUp (int[] t, int start, int stop) {
    int result;
    result = 0;
    for (int i = start; i <= stop; i++) {
        result = result + t [i];
    }
    return result;
}

static void Poker (int[] control, int[] victim1, int[] victim2) {
    int pos, inter;
    for (int i = 0; i < control.Length; i++) {
        pos = control [i];
        inter = victim1 [pos];
        victim1 [pos] = victim2 [pos];
        victim2 [pos] = inter;
    }
}
```

a. When it reaches POINT 1, what will the contents of t1 and t2 be?

b. When it reaches POINT 2, what will the contents of t1, t2 and t3 be? What will the values of ad1 and ad2 be?

Solutions

1.

Sum = 0;

t:

0	1	0	1	0	1	0
0	1	2	3	4	5	6

2.

Sum = 10;

t:

0	2	2	6	4
0	1	2	3	4

3.

t1:

0	0	2	0	4	0	0	6	0	0
0	1	2	3	4	5	6	7	8	9

t2:

1	2	4	7
0	1	2	3

4.

POINT 1

arrayOfDoubles:

0	10	20	30	40	50	60	70
0	1	2	3	4	5	6	7

arrayOfInts:

1	2	4	7
0	1	2	3

POINT 2

arrayOfDoubles:

0	11	19	30	41	50	60	69
0	1	2	3	4	5	6	7

arrayOfInts:

1	2	4	7
0	1	2	3

5.

a)

vector:

0	3	6	9
0	1	2	3

b) Nothing. An error occurs

6.

POINT 1

t1:

0	2	4	6	8	10	12	14	16	18
0	1	2	3	4	5	6	7	8	9

t2:

0	12	24	36	48	60
0	1	2	3	4	5

POINT 2

t1:

0	2	24	6	48	10	12	14	16	18
0	1	2	3	4	5	6	7	8	9

t2:

0	12	4	36	8	60
0	1	2	3	4	5

t3:

0	2	4
0	1	2

ad1 = 132
ad2 = 120

"Writing" exercises (one-dimensional) FIRST STEPS

Solutions at the end of the section

7. Complete the given program, following the instructions given as comments.

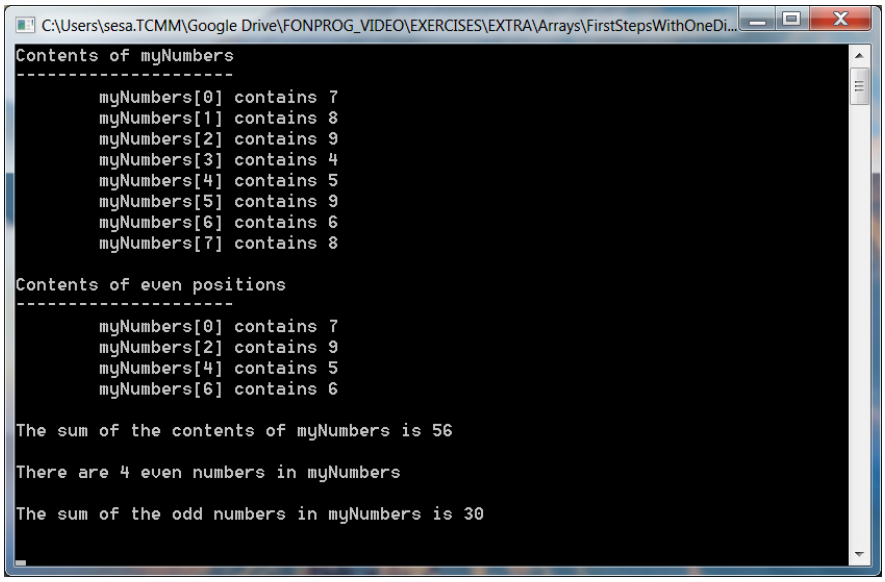
```
public static void Main(string[] args)
{
        int[] myFirstArray;

        // 1: assign to myFirstArray a newly created array with 3 positions

        // 2:   put 1 in position 0
        //      put 10 in position 1
        //      put, in position 2, the sum of the contents of position 0 and 1


        // 3: show (Console.WriteLine...) the contents of position 2

        Console.ReadKey(true);
}
```

8. Complete the given program, following the instructions given as comments.

```
public static void Main(string[] args)
{

        String[] someNames; // declaration

        someNames = new String[5]; // creation

        // let's fill the array referenced by someNames
        someNames[0] = "Alba Artero Amela";
        someNames[1] = "Bernat Benaigues Bel";
        someNames[2] = "Carles Cortada Castro";
        someNames[3] = "Diana Dalmau Duarte";
        someNames[4] = "Esther Eroles Esplandiu";

        // 1: show (Console.WriteLine...) the contents of position 0

        // 2: show (Console.WriteLine...) the contents of position 4

        // 3: try to show the contents of position 5. What happens?

        Console.ReadKey(true);
}
```

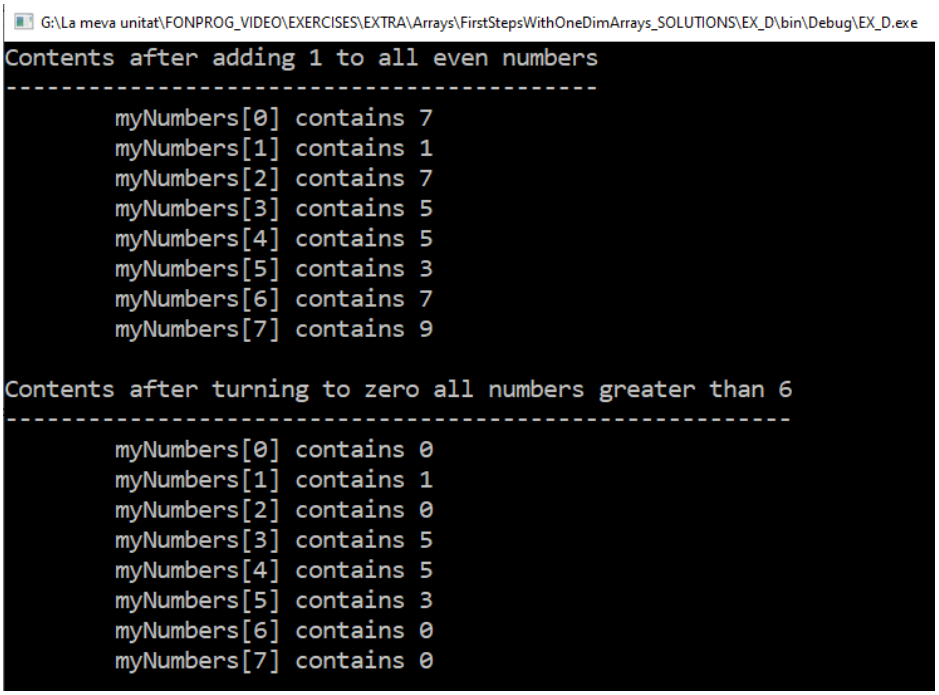
9. Complete the given program, following the instructions given as comments.

```
public static void Main(string[] args)
{
        // declare, create and fill, all in one step
        int[] myNumbers = {7, 8, 9, 4, 5, 9, 6, 8};

        // 1: show contents of myNumbers. Use a for-based iteration

        // 2: show numbers in even positions

        // 3: sum all numbers in myNumbers. Show the result

        // 4: count how many even numbers there are in myNumbers

        // 5: sum the odd numbers in myNumbers

        Console.ReadKey(true);
}
```



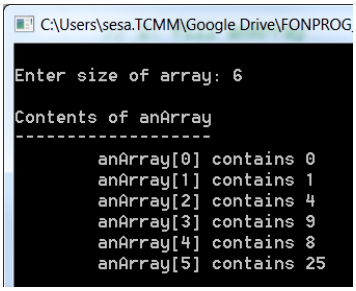
10. Complete the given program, following the instructions given as comments.

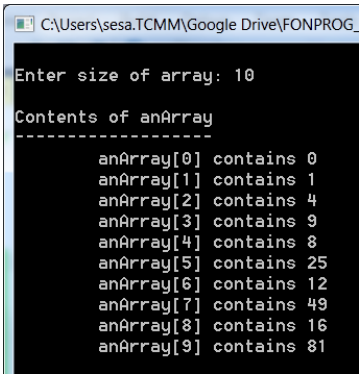
```
public static void Main(string[] args)
{
        // declare, create and fill, all in one step
        int[] myNumbers = { 7, 1, 6, 4, 5, 2, 6, 8 };

        // 1: add one to all even numbers in myNumbers. Show the contents...

        // 2: turn to zero all numbers greater than 6. Show the contents...

        Console.ReadKey(true);
}
```



11. Complete the given program, following the instructions given as comments.

```
public static void Main(string[] args)
{
        int size;
        int[] anArray;

        Console.WriteLine();
        Console.Write("Enter size of array: ");
        size = Convert.ToInt32(Console.ReadLine());

        // 1: assign to anArray a freshly created array of size ints

        // 2: fill anArray
        //             in odd postions put the square of the index
        //             in even positions put the double of the index

        // 3: show the contents of anArray

        Console.ReadKey(true);
}
```





```
// Exercise 7. solution
public static void Main(string[] args)
{
    int[] myFirstArray;

    // 1: assign to myFirstArray a newly created array with 3 positions
    myFirstArray = new int[3];

    // 2:   put 1 in position 0
    //      put 10 in position 1
    //      put, in position 2, the sum of the contents of position 0 and 1
    myFirstArray[0] = 1;
    myFirstArray[1] = 10;
    myFirstArray[2] = myFirstArray[0] + myFirstArray[1];

    // 3: show (Console.WriteLine...) the contents of position 2
    Console.WriteLine("myFirstArray[2] contains {0}", myFirstArray[2]);

    Console.ReadKey(true);
}
```

```
// Exercise 8. Solution
public static void Main(string[] args)
{

    String[] someNames; // declaration

    someNames = new String[5]; // creation

    // let's fill the array referenced by someNames
    someNames[0] = "Alba Artero Amela";
    someNames[1] = "Bernat Benaigues Bel";
    someNames[2] = "Carles Cortada Castro";
    someNames[3] = "Diana Dalmau Duarte";
    someNames[4] = "Esther Eroles Esplandiu";

    // 1: show (Console.WriteLine...) the contents of position 0
    Console.WriteLine("value in position 0: {0}", someNames[0]);

    // 2: show (Console.WriteLine...) the contents of position 4
    Console.WriteLine("value in position 4: {0}", someNames[4]);

    // 3: try to show the contents of position 5. What happens?
    Console.WriteLine("value in position 5: {0}", someNames[5]);
    /*
     * A System.IndexOutOfRangeException makes the program CRASH
     */

    Console.ReadKey(true);
}
```

```
// Exercise 9. Solution
public static void Main(string[] args)
{
    // declare, create and fill, all in one step
    int[] myNumbers = {7, 8, 9, 4, 5, 9, 6, 8};

    // 1: show contents of myNumbers. Use a for-based iteration
    Console.WriteLine("Contents of myNumbers");
    Console.WriteLine("----------------------");
    for (int i=0; i<myNumbers.Length; i++)
    {
        Console.WriteLine("\tmyNumbers[{0}] contains {1}", i, myNumbers[i]);
    }
    Console.WriteLine();

    // 2: show numbers in even positions
    Console.WriteLine("Contents of even positions");
    Console.WriteLine("----------------------");
    for (int i = 0; i < myNumbers.Length; i=i+2)
    {
        Console.WriteLine("\tmyNumbers[{0}] contains {1}", i, myNumbers[i]);
    }
    Console.WriteLine();

    // 3: sum all numbers in myNumbers. Show the result
    int sum = 0;
    for (int i=0; i<myNumbers.Length; i++)
    {
        sum = sum + myNumbers[i];
    }
    Console.WriteLine("The sum of the contents of myNumbers is {0}", sum);
    Console.WriteLine();

    // 4: count how many even numbers there are in myNumbers
    int even = 0;
    for (int i = 0; i < myNumbers.Length; i++)
    {
        if (myNumbers[i] % 2 == 0) even++;
    }
    Console.WriteLine("There are {0} even numbers in myNumbers", even);
    Console.WriteLine();

    // 5: sum the odd numbers in myNumbers
    int sumOdds = 0;
    for (int i = 0; i < myNumbers.Length; i++)
    {
        if (myNumbers[i] % 2 == 1) sumOdds += myNumbers[i];
    }
    Console.WriteLine("The sum of the odd numbers in myNumbers is {0}", sumOdds);
    Console.WriteLine();

    Console.ReadKey(true);
}
```

```
// Exercise 10. Solution
public static void Main(string[] args)
{
    // declare, create and fill, all in one step
    int[] myNumbers = { 7, 1, 6, 4, 5, 2, 6, 8 };

    // 1: add one to all even numbers in myNumbers. Show the contents...
    for (int i=0; i<myNumbers.Length; i++)
    {
        if (myNumbers[i] % 2 == 0) myNumbers[i]++;
    }
    Console.WriteLine("Contents after adding 1 to all even numbers");
    Console.WriteLine("-----------------------------------------");
    for (int i=0; i<myNumbers.Length; i++)
    {
        Console.WriteLine("\tmyNumbers[{0}] contains {1}", i, myNumbers[i]);
    }
    Console.WriteLine();

    // 2: turn to zero all numbers greater than 6. Show the contents...
    for (int i = 0; i < myNumbers.Length; i++)
    {
        if (myNumbers[i]>6) myNumbers[i]=0;
    }
    Console.WriteLine("Contents after turning to zero all numbers greater than 6");
    Console.WriteLine("-------------------------------------------------------");
    for (int i = 0; i < myNumbers.Length; i++)
    {
        Console.WriteLine("\tmyNumbers[{0}] contains {1}", i, myNumbers[i]);
    }
    Console.WriteLine();

    Console.ReadKey(true);
}
```

```
// Exercise 11. Solution
public static void Main(string[] args)
{
    int size;
    int[] anArray;

    Console.WriteLine();
    Console.Write("Enter size of array: ");
    size = Convert.ToInt32(Console.ReadLine());

    // 1: assign to anArray an freshly created array of size ints
    anArray = new int[size];

    // 2: fill anArray
    //      in odd postions put the square of the index
    //      in even positions put the double of the index
    for (int i=0; i<anArray.Length; i++)
    {
        if (i % 2 == 0) anArray[i] = 2 * i;
        else anArray[i] = i * i;
    }

    // 3: show the contents of anArray
    Console.WriteLine("\nContents of anArray");
    Console.WriteLine("-------------------");
    for (int i=0; i<anArray.Length; i++)
    {
        Console.WriteLine("\tanArray[{0}] contains {1}", i, anArray[i]);
    }

    Console.ReadKey(true);
}
```

"Writing" exercises (one-dimensional) OTHER

Solutions at the end of the section

12. Write a function that given two arrays of doubles (assume they have the same length) computes its scalar product. Remember that the scalar product of (x_0, x_1, ..., x_n) and (y_0, y_1... y_n) is $x_0y_0 + x_1y_1 + ... + x_ny_n$. Name your function ScalarProduct.

    Use the following program (or a similar one) to test the function

```
public static void Main (string[] args)
{
    double sp;

    double[] v1 = {8.0, 6.3, 4.1, 3.2, 8.7 };
    double[] v2 = {3.3, -6.1, 8.9, -9.9, 0.3 };

    sp = ScalarProduct (v1, v2);

    Console.WriteLine ("The scalar product of ");
    ShowArray (v1,false);
    Console.WriteLine("and");
    ShowArray (v2, false);
    Console.WriteLine ("is {0}", sp);

    Console.ReadKey (true);
}

static void ShowArray (double[] anArray, bool vertical) {
    // show all the elements of the array
    if (vertical) {
        for (int i = 0; i < anArray.Length; i++) {
            Console.WriteLine ("[{0}]: \t{1}", i, anArray[i]);
        }
    } else {
        for (int i = 0; i < anArray.Length - 1; i++) {
            Console.Write ("{0}, ", anArray[i]);
        }
        Console.WriteLine ("{0}. ", anArray[anArray.Length-1]);
    }
}
```

13. Write a function that given an array of integers and an integer value, counts how many times the value appears in the array. Assume the array is fully initialized.

14. Write a function that given an array of integers counts how many times the last value (the value occupying the last position) appears in the array. Do not count the last value itself. Assume the array is fully initialized. Also write a short program to test your function.

```
In array
1, 2, 3, 1, 2, 3, 0, 0, 1.

The last value appears 2 times


In array
2, 5, 6, 8, 9, 7.

The last value appears 0 times
```

15. Write a function that given an array of integers computes the sum of all the positions the values of which are strictly greater than the value occupying the last position. Assume the array is fully initialized. Also write a short program to test your function

```
In array
1, 2, 3, 8, 9, 10, 7.

The sum of the values greater than the last is 27


In array
1, 2, 3, 4, 5, 6, 7.

The sum of the values greater than the last is 0
```

16. Write a function that given two arrays of integers (assume they have the same length and are fully initialized) determines whether they are identical or no. Hint: try to find a position with different values. If the search succeeds the arrays are not identical. If the search fails then the arrays are identical. Write a program to test your function.

```
Arrays
1, 2, 3, 4, 5, 6, 7.
and
1, 2, 3, 4, 5, 6, 7.
Are they identical? True


Arrays
1, 2, 3, 4, 5, 6, 7.
and
1, 2, 3, 4, 5, 6, 8.
Are they identical? False


Arrays
1, 2, 3, 4, 5, 6, 7.
and
1, 9, 3, 4, 5, 6, 7.
Are they identical? False
```

17. Write a function that given a fully initialized array of doubles and a double value (v) determines whether the value appears in the array or not. Write a program to test your function.

```
In array
1, 2.5, 1.5, 2, 0, 1, -1.

Does the value -1 appear in it? True
Does the value -0.5 appear in it? False
Does the value 0 appear in it? True
Does the value 0.5 appear in it? False
Does the value 1 appear in it? True
Does the value 1.5 appear in it? True
Does the value 2 appear in it? True
Does the value 2.5 appear in it? True
Does the value 3 appear in it? False
```

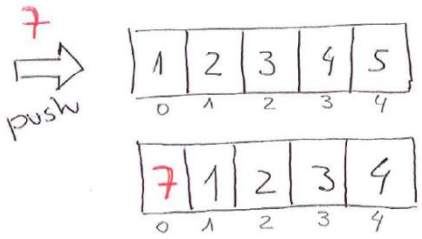
18. Same as before but now returning the index of the position where the value (first) appears or -1 if the value does not appear in the array (since all positions in a table are numbered from 0 onwards, -1 is a suitable value to return when no position contains the searched value.

```
In array
1, 2, 1.5, 2, 0, 1, -1.

The value -1 APPEARS in position 6
The value -0.5 DOES NOT APPEAR
The value 0 APPEARS in position 4
The value 0.5 DOES NOT APPEAR
The value 1 APPEARS in position 0
The value 1.5 APPEARS in position 2
The value 2 APPEARS in position 1
The value 2.5 DOES NOT APPEAR
The value 3 DOES NOT APPEAR
```

19. Write a function that given two fully initialized arrays of integers (make no assumptions regarding their lengths) determines whether all the values of the first one also appear in the second or not. Assume you already have a function that given an array of integers and an integer value determines if the value appears or not in the array.
`static int SearchElement (int[] anArray, int element)`

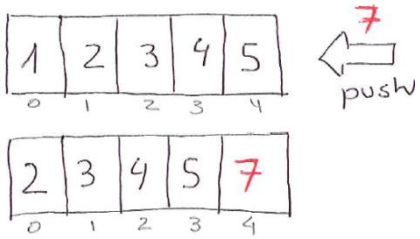
```
Array
1, 2, 3.
is contained in Array
7, 3, 8, 2, 1, 5, 3.
True


Array
1, 2, 3.
is contained in Array
1, 2, 2, 5, 1, 2, 1.
False
```

20. Write a procedure that given a fully initialized array of integers and an integer value "pushes" the value in front of the array making all the values shift a position to the right. The value in the last position is lost.

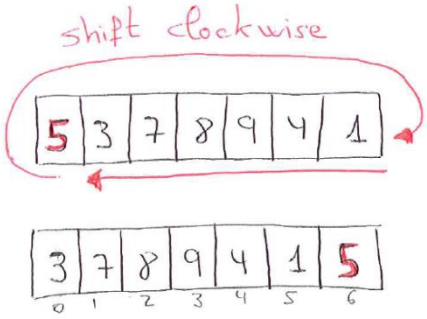



```
Before:
1, 2, 3, 4, 5.


After pushing 7 in front:
7, 1, 2, 3, 4.
```

21. Now the value is pushed at the end of the array, making all the values shift a position to the left. The value in the first position is lost

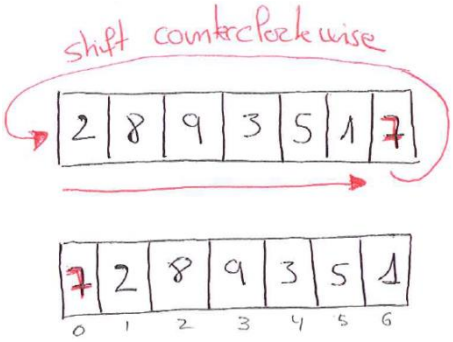


22. Write a procedure that given a fully initialized array of integers rotates it clockwise (the first element goes to the last position and the rest are shifted one position to the left)




```
Before:
5, 3, 7, 8, 9, 4, 1.


After rotating clockwise
3, 7, 8, 9, 4, 1, 5.
```

23. Now the rotation is counterclockwise (the last element goes to the first position and the rest are shifted one position to the right)



24. A polynomial of degree N can be represented with a one-dimensional array of length N+1 with each position storing a coefficient (position 0 stores the coefficient of degree 0, position 1 the coefficient with degree 1, ..., and position N –the last- the coefficient with degree N). Write
    a. A function that given a an array representing a polynomial P(x) and a double value (a) computes P(a). Use Math.Pow to compute powers.
    b. A function that given an array representing a polynomial P(x) returns another array representing its derivative (P'(x))
    c. An action that given an array representing a polynomial "writes" it

    A short program to test your functions is already provided.

```
The polynomial P(x) = 4x^3 + 3x^2 + 1x + 5
Has the following derivative: P'(x) = 12x^2 + 6x + 1
P(0) = 5
P(1) = 13
P(2) = 51
P(3) = 143
```

```
// Exercise 12. Scalar product function.
static double ScalarProduct (double[] x, double[] y) {
    double sum = 0;
    for (int i = 0; i < x.Length; i++) {
        sum = sum + x [i] * y [i];
    }
    return sum;
}
```

```
// Exercise 13. Counting function. Solution
static int CountElement (int [] array, int value) {
    int count = 0;
    for (int i = 0; i < array.Length; i++) {
        if (array [i] == value) {
            count++;
        }
    }
    return count;
}
```

```
// Exercise 14. Count apparitions of last element. solution
static int CountTheLast (int [] array) {
    int count = 0;
    for (int i = 0; i < array.Length-1; i++) { // don't count the last itself
        if (array [i] == array[array.Length-1]) {
            count++;
        }
    }
    return count;
}
```

```
// Exercise 15. Solution
static int SumOfGreaterThanTheLast (int [] array) {
    int sum = 0;
    for (int i = 0; i < array.Length-1; i++) { // discard the last itself
        if (array [i] > array[array.Length-1]) {
            sum = sum + array [i];
        }
    }
    return sum;
}
```

```
// Exercise 16. Array equality. Solution
static bool EqualArrays (int[] x, int[] y) {
    // x and y have the same length
    for (int i = 0; i < x.Length; i++) {
        if (x [i] != y [i]) {
            // discrepancy found: they're not equal
            return false;
        }
    }
    // if here, no discrepancy has been found. They're equal
    return true;
}
```

```
// Exercise 17. Solution
static bool ValueInArray (double [] a, double val) {
    for (int i = 0; i < a.Length; i++) {
        if (a [i] == val)
            return true;
    }
    return false;
}
```

```
// Exercise 18. Solution
static int FindValue(double [] a, double val) {
    for (int i = 0; i < a.Length; i++) {
        if (a [i] == val)
            return i;
    }
    return -1;
}
```

```
// Exercise 19. Solution
static bool Included (int[] first, int[] second) {
    for (int i = 0; i < first.Length; i++) {
        if (SearchElement (second, first [i]) == -1) {
            // first[i] does not appear in second. No need to go on checking
            return false;
        }
    }
    // if here, all the elements in first also appear in second
    return true;
}
```

```
// Exercise 20. Solution
static void PushInFront (int [] array, int value) {
    // shift all elements one position to the right
    for (int i = array.Length - 2; i >= 0; i--) {
        array [i + 1] = array [i];
    }
    // put value in front
    array [0] = value;
}
```

```
// Exercise 21. Solution
static void PushAtTheEnd (int [] array, int value) {
    // shift all elements one position to the left
    for (int i = 1; i < array.Length; i++) {
        array [i - 1] = array [i];
    }
    // put value at the end
    array [array.Length - 1] = value;
}
```

```
// Exercise 22. Solution
static void RotateClockwise (int[] array)
{

    int first;
    // save the first
    first = array [0];
    // shift all elements one position to the left (as in PushAtTheEnd)
    for (int i = 1; i < array.Length; i++) {
        array [i - 1] = array [i];
    }
    // put first at the end
    array [array.Length - 1] = first;
}
```

```
// Exercise 23. Solution
static void RotateCounterClockwise (int [] array) {

    int last;
    // save the last
    last = array [array.Length - 1];
    // shift all elements one position to the right (as in PushInFront)
    for (int i = array.Length - 2; i >= 0; i--) {
        array [i + 1] = array [i];
    }
    // put last in front
    array [0] = last;
}
```

```
// Exercise 24. Solution
// This solution comprises three procedures

static double Evaluate (double [] poli, double a) {
    double result = 0;
    for (int exp = 0; exp < poli.Length; exp++) {
        result = result + poli[exp]*Math.Pow (a, exp);
    }
    return result;
}

static double [] ComputeDerivative (double [] poli) {
    double[] derivative = new double[poli.Length - 1];
    for (int i = 1; i < poli.Length; i++) {
        derivative [i - 1] = poli [i] * i;
    }
    return derivative;
}

static void showPoli (double [] poli, string name) {
    Console.Write (name + "(x) = ");
    for (int exp = poli.Length - 1; exp >= 2; exp--) {
        Console.Write (poli[exp]+"x^"+exp+" + ");
    }
    Console.Write (poli[1]+"x + "+poli[0]);
}
```

OTHER EXERCISES (One-dimensional)

Exercise "Parity Stability"

a) Write **a function** that given a vector of int values determines whether it is parity-stable or not. A vector is said to be parity-stable if all its positions contain a value the parity of which is the same as the parity of the index (odd positions, 1, 3, …, contain odd values and even positions, 0, 2, … contain even values)
For instance, the vector

20	51	30	19	22	1
0	1	2	3	4	5

IS parity-stable since each position contains a value with the same parity as the parity of its index, while the vector

22	73	17	15	6
0	1	2	3	4

IS NOT parity-stable because not all positions contain a value with the same parity as the parity of their indexs (position 2 –even- contains 17 –odd-).

b) Write **a program** that tests the function of the previous part with the two vectors given as examples. The program writes stable if the function finds the vector parity-stable and non-stable otherwise. The followin image shows the expected result

```
Parity Stability
------ ---------

This vector: 20, 51, 30, 19, 22, 1.  is stable

This other vector: 22, 73, 17, 15, 6.  is non-stable
```

Solution on next page

```
// Parity Stability. Solution
static bool IsPartityStable (int [] vector)
{
    // this is a SEARCH. Find a position where parity equality
    // does not old. If such a position is found, vector is
    // not stable. Otherwise vector is stable

    for (int idx = 0; idx<vector.Length; idx++)
    {
        if (vector[idx]%2!=idx%2)
        {
            return false;
        }
    }
    // if here nothing has been found
    return true;
}


// main
public static void Main (String  [] args) {
    int[] testVector_1 = { 20, 51, 30, 19, 22, 1};
    int[] testVector_2 = {22, 73, 17, 15, 6};

    Console.WriteLine ("Parity Stability");
    Console.WriteLine ("------ ---------");
    Console.WriteLine ();

    /* COMPLETE */
    Console.Write("This vector: ");
    WriteVector(testVector_1);
    if (IsPartityStable(testVector_1))
    {
        Console.WriteLine(" is stable");
    }
    else
    {
        Console.WriteLine(" is non stable");
    }

    Console.WriteLine();
    Console.Write("This other vector: ");
    WriteVector(testVector_2);
    if (IsPartityStable(testVector_2))
    {
        Console.WriteLine(" is stable");
    }
    else
    {
        Console.WriteLine(" is non stable");
    }
```

“Writing” exercises (two-dimensional)

Some solutions at the end of the section

25. Write a function that given a fully initialized integer matrix (a matrix of integers) and an integer value r, calculates the sum of the elements in the r^{th} row of the matrix.

26. Write a function that given a fully initialized integer matrix and an integer value c, calculates the sum of the elements in the c^{th} column of the matrix

27. Write a function that given a fully initialized square matrix of integers calculates the sum of the elements in its main diagonal (the one that goes from the upper-left corner to the bottom-right corner)

28. Write a function that given a fully initialized square matrix of integers calculates the sum of the elements in its counterdiagonal (the one that goes from the upper-right corner to the bottom-left corner).

29. A magic matrix is a square matrix in which all rows, all columns and both diagonals add up to the same number. Write a function that given a square matrix determines whether it is magic or not. Hint: calculate the sum of one of the diagonals and then search for an element (column, row or the other diagonal) with a different sum. If such an element is found the matrix is not magic, otherwise it is magic. The following is an example of a 5x5 magic matrix

23	6	19	2	15
10	18	1	14	22
17	5	13	21	9
4	12	25	8	16
11	24	7	20	3

You may find the following program useful for testing your function(s):

```
public static void Main (string[] args)
{
    int[,] aMatrix =     {{23, 6, 19, 2, 15 },
                          {10, 18, 1, 14, 22},
                          {17, 5, 13, 21, 9},
                          {4, 12, 25, 8, 16},
                          {11, 24, 7, 20, 3}};

    Console.WriteLine ("\nThe matrix\n");
    ShowTable (aMatrix);
    if (MagigMatrix (aMatrix)) {
        Console.WriteLine ("\nIS MAGIC !!!");
    } else {
        Console.WriteLine ("\nis not  magic !!!");
    }

    Console.ReadKey ();
}

static void ShowTable (int[,] t) {
    for (int r = 0; r < t.GetLength (0); r++) {
        for (int c = 0; c < t.GetLength (1); c++) {
            Console.Write (t[r, c]+" ");
        }
        Console.WriteLine ();
    }
}
```

30. Complete the following program that determines whether a two-dimensional matrix has a row with more even numbers than odd numbers. If so, show the index of one such row.

```
public static void Main(string[] args)
{
    int row;

    int[,] matrix = {
        {7, 3, 5, 6, 1, 4, 1}, //2
        {6, 2, 4, 3, 5, 3, 1}, //3
        {2, 4, 6, 7, 8, 9, 6}, //5  <==
        {7, 6, 4, 2, 6, 2, 2}, //6  <==
        {1, 5, 5, 8, 7, 0, 5}, //2
        {9, 5, 7, 3, 3, 6, 9}, //1
    //   2  3  3  3  2  4  2
    };

    Console.WriteLine("In matrix: ");
    ShowMatrix(matrix);

    /* TODO 3: complete */

    Console.ReadKey();

} // end of main


static bool MoreEvenThanOdd(int[,] m, int row)
{
    // PROCESS A SINGLE ROW

    /* TODO 1: this functions determines whether row
        (second parameter) has more even numbers than odd numbers
        or not.
     */
}

static int HasRowWithMoreEven(int[,] mat)
{
    // does mat has a row with more even numbers than odd numbers?

    /* TODO 2: return the index of the first row
        with more even numbers than odd numbers. Return -1 if no such
        row exists. YES, this is a SEARCH
     */
}



// complimentary procedure
static void ShowMatrix(int[,] mat)
{
    Console.ForegroundColor = ConsoleColor.Yellow;
    Console.WriteLine();
    for (int i = 0; i < mat.GetLength(0); i++)
    {
        for (int j = 0; j < mat.GetLength(1); j++)
        {
            Console.Write(mat[i,j] + " ");
        }
        Console.WriteLine();
    }
    Console.WriteLine();
    Console.ResetColor();
}
```

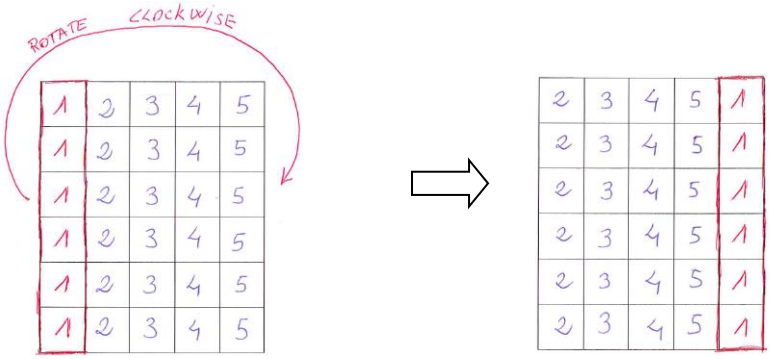
```
In matrix:

7 3 5 6 1 4 1
6 2 4 3 5 3 1
2 4 6 7 8 9 6
7 6 4 2 6 2 2
1 5 5 8 7 0 5
9 5 7 3 3 6 9

There's a row with more even numbers: 2
```

31. Complete the following program that, in a two-dimensional matrix, finds the column that contains more even numbers. The program shows the index of that column

```
public static void Main(string[] args)
{
    int maxCol, maxEven;

    int[, ] matrix = {
        {7, 3, 5, 6, 1, 4, 1}, //2
        {6, 2, 4, 3, 5, 3, 1}, //3
        {2, 4, 6, 7, 8, 9, 6}, //5
        {7, 6, 4, 2, 6, 2, 2}, //6
        {1, 5, 5, 8, 7, 0, 5}, //2
        {9, 5, 7, 3, 3, 6, 9}, //1
    //   2  3  3  3  2  4  2
    //                  ^
    };

    Console.WriteLine("In matrix: ");
    ShowMatrix(matrix);

    maxCol = ColumnWithMoreEven(matrix);
    maxEven = CountEvenInColumn(matrix, maxCol);

    Console.WriteLine("The column with more even numbers has index: " + maxCol);
    Console.WriteLine("And it contains " + maxEven + " even numbers");

    Console.ReadKey();
} // main ends here

static int CountEvenInColumn(int[,] m, int col)
{
    // PROCESS A SINGLE COLUMN

    /* TODO 1: count the occurrences of even numbers
     * in the specified column */
}

static int ColumnWithMoreEven(int[,] m)
{
    // this is a "FIND THE MAXIMUM" problem

    /* TODO 2: return the index of the column with
        more even numbers
     */
}

// complimentary procedure
static void ShowMatrix(int[,] mat)
{
    Console.ForegroundColor = ConsoleColor.Yellow;
    Console.WriteLine();
    for (int i = 0; i < mat.GetLength(0); i++)
    {
        for (int j = 0; j < mat.GetLength(1); j++)
        {
            Console.Write(mat[i, j] + " ");
        }
        Console.WriteLine();
    }
    Console.WriteLine();
    Console.ResetColor();
}
```

```
In matrix:

7 3 5 6 1 4 1
6 2 4 3 5 3 1
2 4 6 7 8 9 6
7 6 4 2 6 2 2
1 5 5 8 7 0 5
9 5 7 3 3 6 9

The column with more even numbers has index: 5
And it contains 4 even numbers
```

32. Write a procedure that given a matrix of integers (int [,]) rotates its columns clockwise.

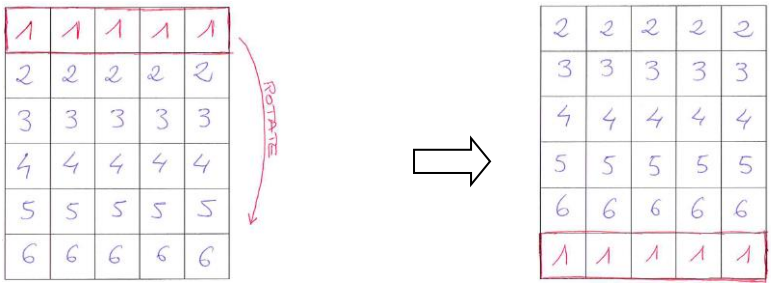


Use the following program (or a similar one) to test the procedure

```
public static void Main (string[] args)
{
    int[,] myMatrix = {{1, 2, 3, 4, 5, 6},
      {1, 2, 3, 4, 5, 6},
      {1, 2, 3, 4, 5, 6},
      {1, 2, 3, 4, 5, 6},
      {1, 2, 3, 4, 5, 6},
      {1, 2, 3, 4, 5, 6},
      {1, 2, 3, 4, 5, 6},
      {1, 2, 3, 4, 5, 6},};

    ConsoleKeyInfo cki;
    bool stop = false;

    WriteMatrix (myMatrix,1,1);

    while (!stop) {
        cki = Console.ReadKey (false);
        switch (cki.Key) {
        case ConsoleKey.Spacebar:
            RotateClokwise (myMatrix);
            WriteMatrix (myMatrix,1,1);
            break;
        case ConsoleKey.Q:
            stop = false;
            break;
        }
    }
}

static void WriteMatrix (int[,] matrix, int ve, int ho) {
    // ve: vertical offset (initial row)
    // ho: horizontal offset (initial column)

    for (int row = 0; row < matrix.GetLength (0); row++) {
        // position at the right place
        Console.SetCursorPosition (ho, row+ve);
        for (int column = 0; column < matrix.GetLength (1); column++) {
            Console.Write (matrix[row, column]);
        }
    }
}
```

33. Id. Counterclockwise

34. Write a procedure that given a matrix of integers (int [,]) rotates its rows top-down.



35. Id bottom-up.

```
// Exercise 25. Solution
static int SumRow (int[,] matrix, int r) {
    int sum = 0;

    // iterate over columns. Fix row (it's always r)
    for (int i = 0; i < matrix.GetLongLength (1); i++) {
        sum = sum + matrix [r, i];
    }
    return sum;
}
```

```
// Exercise 26. Solution
static int SumColumn (int[,] matrix, int c) {
    int sum = 0;

    // iterate over rows. Fix column (it's always c)
    for (int i = 0; i < matrix.GetLongLength (1); i++) {
        sum = sum + matrix [i, c];
    }
    return sum;
}
```

```
// Exercise 27. Solution
static int SumMainDiagonal (int[,] squareMatrix) {
    int sum = 0;

    for (int i = 0; i < squareMatrix.GetLength (0); i++) {
        // being square
        // squareMatrix.GetLength (0) == squareMatrix.GetLength (1)

        sum = sum + squareMatrix [i, i];
    }
    return sum;
}
```

```
// Exercise 28. Solution
static int SumCounterDiagonal (int[,] squareMatrix) {
    int sum = 0;
    int c;
    int SIZE = squareMatrix.GetLength (0);

    // iterate over rows. For each row determine column (c)
    for (int r = 0; r < SIZE; r++) {
        c = SIZE - 1 - r;
        sum = sum + squareMatrix[r,c];
    }
    return sum;

    /* shorter version

        int sum = 0;

        for (int r=0; r<squareMatrix.GetLength(0); r++) {
            sum = sum + squareMatrix[r, squareMatrix.GetLongLength(0)-1-r];
        }
        return sum;

    */
}
```

```
// Exercise 29. Solution
static bool MagigMatrix (int[,] squareMatrix) {
    int value;

    value = SumMainDiagonal (squareMatrix);

    // check counter diagonal
    if (SumCounterDiagonal(squareMatrix)!=value) {
        // discrepancy found
        return false;
    }
    else {
        // both diagonals add up to the same value

        // let's check the rows
        for (int row = 0; row < squareMatrix.GetLength (0); row++) {
            if (SumRow (squareMatrix, row) != value) {
                // discrepancy found
                return false;
            }
        }
        // if here, no discrepancy have been found in rows
        // let's check columns
        for (int column = 0; column < squareMatrix.GetLength (0); column++) {
            if (SumColumn (squareMatrix, column) != value) {
                // discrepancy found
                return false;
            }
        }
        // if here, not discrepancy at all has been found: Matrix IS MAGIG
        return true;

    } // end of else
}
```

```
// Exercise 30. Solution
public static void Main(string[] args)
{
    int row;

    int[,] matrix = {
        {7, 3, 5, 6, 1, 4, 1}, //2
        {6, 2, 4, 3, 5, 3, 1}, //3
        {2, 4, 6, 7, 8, 9, 6}, //5  <==
        {7, 6, 4, 2, 6, 2, 2}, //6  <==
        {1, 5, 5, 8, 7, 0, 5}, //2
        {9, 5, 7, 3, 3, 6, 9}, //1
    //   2  3  3  3  2  4  2
    };

    Console.WriteLine("In matrix: ");
    ShowMatrix(matrix);

    row = HasRowWithMoreEven(matrix);
    if (row == -1)
    {
        Console.WriteLine("There's no row with more even numbers");
    }
    else {
        Console.WriteLine("There's a row with more even numbers: " + row);
    }

    Console.ReadKey();

} // end of main
```

```
static bool MoreEvenThanOdd(int[,] m, int row)
{
    // process a a row

    int even, odd;

    even = 0;
    odd = 0;

    for (int col = 0; col < m.GetLength(1); col++)
    {
        if (m[row,col] % 2 == 0)
        {
            even++;
        }
        else {
            odd++;
        }
    }

    return even > odd;
}

static int HasRowWithMoreEven(int[,] mat)
{
    // does mat has a row with more even numbers than odd numbers?

    // THIS IS A SEARCH
    for (int row = 0; row < mat.GetLength(0); row++)
    {
        if (MoreEvenThanOdd(mat, row)) return row;
    }
    return -1;
}
```

```
// Exercise 31. Solution

static int CountEvenInColumn(int[,] m, int col)
{
    // to process a column fix it (fix its index) and and process all its rows

    int counter = 0;

    for (int row = 0; row < m.GetLength(0); row++)
    { // iterate over rows
        if (m[row,col] % 2 == 0) counter++;
    }

    return counter;
}

static int ColumnWithMoreEven(int[,] m)
{
    // this is a "find the maximum" problem

    int colMaxSoFar = 0;
    int maxSoFar = CountEvenInColumn(m, colMaxSoFar);
    int count;

    for (int col = 1; col < m.GetLength(1); col++)
    {
        count = CountEvenInColumn(m, col);
        if (count > maxSoFar)
        {
            maxSoFar = count;
            colMaxSoFar = col;
        }
    }

    return colMaxSoFar;
}
```

```
// Exercise 32. Solution
static void RotateMatrixColumnClokwise (int [,] matrix) {
    int[] leftmostColumn;

    leftmostColumn = new int[matrix.GetLength (0)];

    // make a copy of the leftmost column
    for (int r = 0; r < matrix.GetLength (0); r++) {
        leftmostColumn [r] = matrix [r, 0];
    }

    // move cols from 1 to last one position to the left
    for (int c = 1; c < matrix.GetLength (1); c++) {
        for (int r = 0; r < matrix.GetLength (0); r++) {
            matrix [r, c - 1] = matrix [r, c];
        }
    }

    // put the copy in the rightmost column of the matrix
    for (int r = 0; r < matrix.GetLength (0); r++) {
        matrix [r, matrix.GetLongLength(1)-1] = leftmostColumn [r];
    }
}
```