Exercises that will make you nuts

The goal of this exercises is to think about on how you get to your solution and to keep your solution as maintainable as possible.

Write down tasks. Think about their size (volume of work).

Try to execute.

There is not one solution. The same result can be found using several methods. But it will take a lot of thinking power to find out if you must loop, switch, decide, getting out of the loop, what routine, what size of arrays, you must use.

You will get stuck in your solution and often you will have to decide to recode existing stuff or even restart.

This is the goal of the exercise. You will have to rework your code. How do you react on this? Do you find it fun or annoying?

Exercise 10.01

- You have an array of 5 by 5. (Array has 2 dimensions).

Notes

.....

.....

.....

.....

.....

.....

.....

- You ask through the console a starting position. Meaning, in what column and what row you place the number 1.
- Add checks if it is possible.
  - E.g. Going to row 15 and column 8 is not possible.
  - Re-ask until you got a correct location.
- Let's assume that you want the number 1 in row 1 and column 1. (or 0, 0 if you work with indexes in C# 😊)

1				

- The next number, 2 can be placed at different locations.
- These are the rules.
  - When you go horizontal, you jump over 2 cells.
  - When you go vertical, you jump over 2 cells.
  - When you go diagonal, you jump over 1 cell.
  - You can only place a number in a location if there is no number yet.

Notes

.....

.....

.....

.....

.....

.....

.....

COPY PASTE)

- This means that the next number (in this case 2) can be placed in 3 different locations.

1			**2**	
		2		
2				

- You have to try the direction where you can place the next number, in a specific order.
  - ➔
  - ➘
  - ↓
  - ↙
  - ←
  - ↖
  - ↑
  - ↗
- This means that for the example I have to place the number 2 in the same row as where the 1 was placed, because I first try to go to the direction ➔.

Notes

.....

.....

.....

.....

.....

.....

.....

COPY PASTE)

- Continuing with the existing rules, you can fill in the grid like this.

1	8		2	9
16	13	5	17	14
		10		
4	7	15	3	6
19	12		18	11

- At a certain moment you are blocked. Meaning, you can't put the next number on the grid according to the given rules.
- When that happens, you show the grid to the console.

Example result

```
In what column do you want to put number 1?
6
This is not possible
In what column do you want to put number 1?
1
In what row do you want to put number 1?
10
This is not possible
In what row do you want to put number 1?
0
This is not possible
In what row do you want to put number 1?
1

The end result is
1     8           2       9
16    13    5     17      14
            10
4     7     15    3       6
19    12          18      11
```

Notes

Variant 1

- Create a new project, in the same solution.
- Copy your working code to the new project (another location), and try to update the code where needed.
- The same exercise, but you create an array of a length 25.

If you have made your code maintainable, you should only make some changes in small parts of you code.

Exercise 10.02

We continue building on exercise 10.01, the 2-dimension grid.

- When you are blocked, remove the last placed number, and try to place it on the next possible location.
- In our example, this means, you have put 19 into the direction ←, you remove the number 19 and try it on the next possible move.
- In our specific case, the next possible move is direction ↖.

1	**8**		**2**	**9**
16	**13**	**5**	**17**	**14**
	19	**10**		
4	**7**	**15**	**3**	**6**
	12		**18**	**11**

Notes

- And you continue, with your routine.

1	8		2	9
16	13	5	17	14
22	19	10	23	20
4	7	15	3	6
	12	21	18	11

- And you are blocked again.
- The whole process is repeated, until you can place number 25.
- When you have filled in the complete grid, you show that grid towards the console.

Variant 1

- Adapt the other project, by copying the needed code. Try to update the code where needed.
- Can you keep track of all the needed changes?

If you have made your code maintainable, you should only make some changes in small parts of you code.

Notes

Exercise 10.03

We continue building on exercise 10.02, the 2-dimension grid.

- We add one rule.
- When you have placed number 25, you need to check where number 1 is.
- When you can jump from 25 towards 1, with the same rules, it is acceptable.
    - When you go horizontal or vertical, you jump over 2 cells, and you end on number 1.
    - When you go diagonal, you jump over 1 cell, and you end on number 1.
    - The rule of a not filled value does not count here, because it will be filled with a 1.
- If not, you go back in the process and you continue until you have found a solution where you can jump from 25 towards 1.
- When you have found a solution, you put that grid on the console.

Variant 1

- Adapt the other project, by copying the needed code. Try to update the code where needed.
- Can you keep track of all the needed changes?

Notes

Exercise 10.04

We continue building on exercise 10.03, the 2-dimension grid.

- You find all possible solutions, and you put them on the console.
- Also count the solutions and show that on the console.

Variant 1

- Adapt the other project, by copying the needed code. Try to update the code where needed.
- Can you keep track of all the needed changes?

Exercise 10.05

We continue building on exercise 10.04, the 2-dimention grid.

- You ask to the user how big your square must be. Be reasonable in your try-outs. Grids till 8 should go fast, bigger grids will take some time, but at regular base a solution should be shown on the screen.

Variant 1

- Adapt the other project, by copying the needed code. Try to update the code where needed.
- Can you keep track of all the needed changes?

Notes

Exercise 10.06

We continue building on exercise 10.05, the 2-dimension grid.

- You are not working with a square, but with a rectangle. The sizes of it can be different.
- E.g. The grid has size 6 x 5.

Variant 1

- Adapt the other project, by copying the needed code. Try to update the code where needed.
- Can you keep track of all the needed changes?

Exercise 10.07

We continue building on exercise 10.06, the 2-dimension grid.

- If you change the rules in jumping to the next location, what must be changed in your code?
- If it is built up correctly, you don't have a lot of work.
- Try it out.
  - When you go horizontal, you jump over 3 cells.
  - When you go vertical, you jump over 2 cells.
  - When you go diagonal, you jump over 1 cell.

Variant 1

- Adapt the other project, by copying the needed code. Try to update the code where needed.
- Can you keep track of all the needed changes?

Notes