Exercises that will make you crazy

These exercises will really test your organisation skills in working towards a solution.

Think before you execute.

Exercise 09.01: Finding prime numbers

This exercise will give you the prime numbers between 0 and xxx, borders included.

- Ask on the console a number, e.g., 100.
- Test if it is a number, if not, re-ask.

You can really go big, so choose your datatype carefully, but later changing it in another datatype should not be a problem.

You can start with byte, and later go on to short, int, int32 or int64, long. You don't need decimals.

- When you have a number, create an array of Booleans.
- The size of that array is one bigger than the number you have entered.
  - The reason is, we start counting with 0.
- All values on that array must become true.

This is the starting point, we have an array of x + 1 elements, all with value true.

Notes

.....

.....

.....

.....

.....

.....

.....

Method in finding the prime numbers

The goal is now to give all the numbers that are not prime a false value.

- Element 0 of your array becomes false, because it is not a prime.
- Element 1 of your array becomes false, because it is not a prime.
- Now you create a loop.
- Inside the loop, you do this.
  - Find the next element, that is true (in this case 2)
  - Element 2 stays true. All numbers that are a multiplication of 2, become false (only when they are true).
  - Example: 2 stays true, 4 becomes false, 6 becomes false, and so on, till you reached the end of the array.

Pay attention that you are not trying to change an element that does not exist in your array.

E.g., Element 102 will fail if your array does not contain that element.

- Repeat, but now for element 3 (next element that is true).

You do the same for 5 (because 4 is already put to false).

And so on.

- Repeat, but now for element 5 (next element that is true).
  - Element 4 should be already false.
- When you have done all the numbers, you can exit your loop.
- You are left with an array of all false elements, except the prime numbers.
- Show those prime numbers nicely on the console.

Notes

Notes

First let your routine work, then we will make it faster.

On internet you will find faster algorithms to find prime numbers.

Variant 1

- Ask yourself. At what moment do you exit the loop?

Suppose you want to find all prime numbers between 0 and 100.

Do you loop from 0 till 100 or not?

Ask Vincent or your colleagues for a solution.

Variant 2

- The only even prime number is 2, so why not filling your array with odd numbers.
- The result is than 2 + all other prime numbers that are odd.

You can find a lot of examples and info about this exercise when you look for: The sieve of Eratosthenes.

Pay attention: On internet you find a lot of code in C# and all other program languages that solves this exercise. Even ChatGPT will give you a correct routine.

Create it by yourself. You want to learn something. You do not want to "Copy Paste" without understanding.

Exercise 09.02: Keep your code clear during refactoring

- We are going to loop thru numbers. Again 😊.
- From 1 till let's say 30.
- With a given number, we will do some stuff.
  - When the number is even, we divide by 2.
  - When the number is odd, we multiply by 3 and add 1.
  - We repeat, until we reach the number 1.

An example.

*When you start with 5, you will get 16 (5 * 3 + 1).*

And then you get 8 (16 / 2).

The next numbers are 4, 2 and 1 (here you stop).

So 5 → 16 → 8 → 4 → 2 → 1.

- Find for every number (1 till 30), the highest number you encounter.
- In the example of 5, the highest number you will get is 16.

Example

- When you start with 13.
  - 13 → 40 → 20 → 10 → 5 → 16 → 8 → 4 → 2 → 1.
- So that means, the highest number starting with 13 is 40.

Another example

- When you start with 27.
  - 27 → 82 → 41 → … → 5 → 16 → 8 → 4 → 2 → 1
- So that means, the highest number starting with 27 will be 9.232.

What is asked?

- Put on the console for every number from 1 till 30, the highest number you encounter.

Notes

Example of exercise when you loop till 4.

```
1 --> 4
2 --> 2
3 --> 16
4 --> 4
```

Variant 1

- When you do the routine for number 1.
  - Your will have the numbers 4 → 2 → 1.
- This means, when you need to calculate 2 and 4, you have already done that, while calculating number 1.

	*So while calculating 1, you are also calculating 2 and 4, because you pass those numbers.* *When your starting number is 4. You have to do nothing, because in calculating the starting number 1, you have encountered 4.* *You know that the highest number will be 4, because you are encountering in your temporary result the number 4 and 2. By working this way, you have also the maximum value of 2. Pay attention. This is not 4. You don't pass 4 while calculating 2.*

- Keep track of all those numbers, so you can find the end result faster, because you can skip a lot of loops, because you have already done that.

	*Compare the speed of the result of variant 1 with the original exercise.* *When you don't see a speed difference, try the exercise with a loop from 1 till 1.000.*

Notes

.....

.....

.....

.....

.....

.....

.....

COPY PASTE)

Variant 2

- You are busy in the loop.
- So you have some results.
  - 1 ends up with 4.
    - You passed 4 and 2.
  - 2 ends up with 2.
    - You passed 2.
    - You are not doing that when you have solved variant 1.
  - 3 ends up with 16.
    - You passed 3, 10, 5, 16, 8, 4, 2, 1.
- When you look at the passing numbers for 3, you see that you encounter 4, but 4, you passed that doing number 1. That means that you already know what the highest number is that you encounter when passing 4. That is 4. That number you can compare it with the highest at the moment, that was 16. 16 is higher than the 4, so 16 is the result.

	*You can interrupt your looping, because you have already done that in the looping.* *So while doing 3, you can fill in 10, 5, 16, 8. At 4 you can stop, because that is already filled in.* *Stop your looping when possible.*

	*Compare the result of variant 2 with the original and variant 1.* *When you don't see a speed difference, try the exercise with a loop from 1 till 1.000.*

Notes

.....

.....

.....

.....

.....

.....

.....

COPY PASTE)

Variant 3

- You are finding the numbers from 1 till 30.
- Let's make that array a bit bigger using the console.
- You ask thru the console a number, and re-ask as long that it is not a number.
- Let's say you give 2.500.
- You loop all the numbers from 1 till 2.500.

Can you invent other ways to make your routine as fast as possible?

Or is it fast enough?

Variant 4

- On the console, I want to see this result. The routine is the same, the output is different.

```
Please enter a number.
Abc
Abc is not a number, please try again
2500

2 is the biggest when you start with 2.
4 is the biggest when you start with 1 or 4.
8 is the biggest when you start with 8.
16 is the biggest when you start with 3, 5, 6, 10, 12 or 16.
20 is the biggest when you start with 20.
40 is the biggest when you start with 13, 26 or 40.
… (And so on)
```

Variant 5

- Keep also track of the number of steps you need to get to 1.
  - For 1 is that 3 steps (1 → 4 → 2 → 1).
  - For 27 is that 111 steps.
  - For 63.728.127 is that 949 steps.
- Show the result nicely on the console.

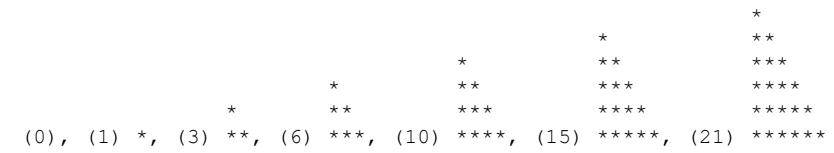
Notes

COPY PASTE)

Exercise 09.03: Counting with triangular numbers

Find the possible combinations.

- Ask on the console a number, e.g., 30.
- Test if it is a number, if not, re-ask.
  - Don't start with a big number.
  - Start small to test your routine.
- Find the triangular numbers.
  - These are the numbers that are needed to form a triangular.
- 0, 1, 3, 6, 10, 15, 21, 28, … till the asked number.
  - Do you see the pattern?



- Every number can be written with the sum of three triangular number.
  - Find all combinations.

- On the console, I want to see this result.

```
Please enter a number.
Abc
Abc is not a number, please try again
6

0 is the sum of 0, 0 and 0. (Solution 1)
1 is the sum of 0, 0 and 1. (Solution 1)
2 is the sum of 0, 1 and 1. (Solution 1)
3 is the sum of 0, 0 and 3. (Solution 1)
3 is the sum of 1, 1 and 1. (Solution 2)
4 is the sum of 0, 1 and 3. (Solution 1)
5 is the sum of 1, 1 and 3. (Solution 1)
6 is the sum of 0, 0 and 6. (Solution 1)
6 is the sum of 0, 3 and 3. (Solution 2)
```

Notes

Exercise 09.04: Walking up the Stairs

Find the number of possible combinations.

- Let us assume you have a stair with 10 steps.
- You can walk up the stairs by using one or two steps at a time.
  - Example 1: You walk up the stairs with 10 one-steps.
  - Example 2: You walk up the stairs with 5 two-steps.
- How many different possible combinations do you have to get to the top?

This is an exercise in your thinking process.

You can calculate all combinations, but you can also find a shortcut.

Start with finding all the combinations and see if you can find the pattern.

Notes

.....

.....

.....

.....

.....

.....

.....

COPY PASTE)

Exercise 09.05: Pascal's Triangle (in another way)

This is an exercise of an array of arrays. You can find a lot of information about Pascal's triangle on the internet.

Please try to create the exercise like I've asked. When you succeed, you will know how to work with arrays. 😊

Part 01

- We will start with creating an array of arrays.
  - The first array has a size of 20.
    - And contains arrays of integers.
  - The first element of the array contains an array of 1 integers.
  - The second element of the array contains an array of 2 integers.
  - …
  - The last element of the array contains an array of 20 integers.
- So you have an array of arrays with different sizes.

Part 02

We will fill the arrays with values (integers).

- Every first element contains a 1.
- Every last element contains a 1.
- Every other element (has an index x) contains the sum of the previous array element with index x and x – 1.
- You should have those values:
  - 1.
  - 1 and 1.
  - 1, 2 and 1.
  - 1, 3, 3 and 1.
  - 1, 4, 6, 4 and 1.

Notes

- The 6 (Third element of the array, is the sum of the third element (a 3) and the second element (also a 3) of the previous array.

Part 03

We will so some calculations.

- Make the sum of all elements of every array and show them on the screen.
  - What do you notice?
- Start with the third array, showing the first item.
  - Continue with the next array, the second item.
  - Continue with the next array, the third item.
  - What do you notice?
    - Look at exercise 09.03.
- Do this calculation on all the arrays.
  - Take the first element of the array.
    - Continue till there is no element or array anymore.
      - Add the next element of the previous array.
    - Show the result.
  - Take the first element of the next array.
    - Continue till there is no element or array anymore.
      - Add the next element of the previous array.
    - Show the result.
  - Repeat till you reached the end of your arrays.
  - What do you notice?

I hope you find this fun 😊

Notes

.....

.....

.....

.....

.....

.....

.....

COPY PASTE)