

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

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

COPY PASTE)

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

COPY PASTE)



*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.

Notes

COPY PASTE)

Exercise 09.02

- 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 $5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$.

- 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.
 - $13 \rightarrow 40 \rightarrow 20 \rightarrow 10 \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$.
- So that means, the highest number starting with 13 is 40.

Another example

- When you start with 27.
 - $27 \rightarrow 82 \rightarrow 41 \rightarrow \dots \rightarrow 5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$
- 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

COPY PASTE)

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.
 - You will have the numbers $4 \rightarrow 2 \rightarrow 1$.
- 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 and 16.
20 is the biggest when you start with 20.
40 is the biggest when you start with 13, 26, 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)