# Work on project. Stage 2/6: A proof of work concept



Project: Blockchain

Hard 12 minutes

## §1. Description



The security of our blockchain is pretty low. You can't just change some information in the middle of a blockchain, because the hash of this block will also be changed. And the next block still keeps the old hash value of the previous block. But can't we replace the old hash value with the new hash value so everything will be ok? No, because when you change the value of the previous hash in the block, the hash of this block will also be changed! To fix this, you need to change the value of the previous hash in the block after it. To solve this problem, you need to fix hash values in all the blocks until the last block of the blockchain!

This seems to be a pretty hard task to execute, doesn't it? If the time it takes to fix the hash value of the previous block is less than time to create a new block, we suddenly would be fixing blocks faster than the system can create them and eventually we will fix them all. The problem is that fixing the hash values is easy to do. The blockchain becomes useless if it is possible to change information in it.

The solution to this is called **proof of work**. This means that creating new blocks and fixing hash values in the existing ones should take time and shouldn't be instant. The time should depend on the amount of computational work put into it. This way, the hacker must have more computational resources than the rest of the computers of the system put together.

The main goal is that the hash of the block shouldn't be random. It should start with some amount of zeros. To achieve that, the block should contain an additional field: a **magic number**. Of course, this number should take part in calculating the hash of this block. With one magic number, and with another, the hashes would be totally different even though the other part of the block stays the same. But with the help of probability theory, we can say that there exist some magic numbers, with which the hash of the block starts with some number of zeros. The only way to find one of them is to make random guesses until we found one of them. For a computer, this means that the only way to find the solution is to brute force it: try 1, 2, 3, and so on. The better solution would be to brute force with random numbers, not with the increasing from 1 to N where N is the solution. You can see this algorithm in the animation below:



Obviously, the more zeros you need at the start of the block hash, the harder this task will become. And finally, if the hacker wants to change some information in the middle of the blockchain, the hash of the modified block would be changed and it won't start with zeros, so the hacker would be forced to find another magic number to create a block with a hash which starts with zeros. Note that the hacker must find magic numbers for all of the blocks until the end of the blockchain, which seems like a pretty impossible task, considering that the blockchain will grow faster.

It's said that that the block is **proved** if it has a hash which starts with some number of zeros. The information inside it is impossible to change even though the information itself is open and easy to edit in the text editor. The result of the edit is a changed hash of the block, no longer containing zeros at the start, so this block suddenly becomes **unproved** after the edit. And since the blockchain must consist of only proved blocks, the whole blockchain becomes invalid. This is the power

of the proof of work concept.

In this stage, you need to improve the blockchain. If should generate new blocks only with hashes that start with N zeros. The number N should be input from the keyboard. Also, the blockchain should be saved to the file after each block. At the start of the program, you should check if a blockchain exists on the hard drive, load it, check if it is valid, and then continue to create blocks. You may want to use serialization to do that.

# §2. Output examples

The example below shows how your output might look. Output information about a few first blocks of the blockchain. Also, output the time that was needed to create a block. Your results and time measurements can be totally different than in the example! To be tested successfully, program should output information about first five blocks of the blockchain. Blocks should be separated by an empty line.

```
Enter how many zeros the hash must starts with: 5

Block:
Id: 1
Timestamp: 1539827383396
Magic number: 24672386
Hash of the previous block:
0
Hash of the block:
00000a3fe20573b5bb358d2291165e15662a5b057240e954c573fb1f2a6d0cb8
Block was generating for 12 seconds

Block:
Id: 2
Timestamp: 1539827385414
Magic number: 87453465
Hash of the previous block:
00000a3fe20573b5bb358d2291165e15662a5b057240e954c573fb1f2a6d0cb8
Hash of the block:
000002e0ddd3c11e85466be0fa3dc5cb112daa7a3126e680c7d4f5716c0c6f9c
Block was generating for 21 seconds

Block:
Id: 3
Timestamp: 1539827387961
Magic number: 32734621
Hash of the previous block:
000002e0ddd3c11e85466be0fa3dc5cb112daa7a3126e680c7d4f5716c0c6f9c
Hash of the block:
000006edc10682ac3d511175b54192a7d36459af6e23671275c2c6879ab1c412
Block was generating for 18 seconds
```

```
Enter how many zeros the hash must starts with: 8

Block:
Id: 1
Timestamp: 1539827504324
Magic number: 9347534
Hash of the previous block:
0
Hash of the block:
0000000031ae66963218b132a7c9e7e6ee300a39288e80ce8f6b107aca6d467b
Block was generating for 231 seconds

Block:
Id: 2
Timestamp: 1539827526140
Magic number: 34652436
Hash of the previous block:
0000000031ae66963218b132a7c9e7e6ee300a39288e80ce8f6b107aca6d467b
Hash of the block:
00000000526655e7dee356b943c5551f0dededd67d0b36db34a3e5d03e44aad6
Block was generating for 211 seconds

Block:
Id: 3
Timestamp: 1539827557451
Magic number: 84587649
Hash of the previous block:
00000000526655e7dee356b943c5551f0dededd67d0b36db34a3e5d03e44aad6
Hash of the block:
00000000df645313e301f147105b009bdc084945fb684517d351f175ed4d67be
Block was generating for 461 seconds
```

Code Editor IDE

Solve in IDE

✓ **IDE is responding** IntelliJ IDEA 2019.3

✓ **Plugin is responding** 3.2-2019.3-3686

This content was created 11 months ago and updated 2 days ago. Share your feedback below in comments to help us improve it!

Comments (11) Hints (0) Useful links (0) Solutions (0)

**Share something, Sergey Kubatko**

Post Please do not post solutions here

Sort by: **Last posted**

MG **Marcin G** 5 days ago Report

Why exactly sould i search for magic numbervia random function? If i try to do 1, 2, 3 etc. i can be sure that i don't miss anu number and don't loose time for redundant numbers...

♡ 0 Reply

LA **LAURENT APICELLA** 3 months ago Report

what is the use of an error message like this one ?
You should output 5 blocks, found 0

Especially when just below there are :
Block:

♡ 0 Show all Reply

CH **Christian H** 4 months ago Report

I get error that each block should have nine lines of data. Mine do and are not being accepted.

♡ 0 Reply

Ak **ali katlabi** 4 months ago Report

I have the  IDE the plugin installed , yet the (solve in IDE) not active , I cannot submit

♡ 0 Reply

U9 **User 997340** 5 months ago Report

Wrong answer in test #2  You should output 5 blocks, found 10.... BUT I HAVE 5 BLOCKS IN IDE

♡ 0 Reply

NB **Nikhil Bhatnagar** 6 months ago Report

where is the magic number animation link?

♡ 0 Reply

RK **Rafał Koszałkowski** 6 months ago Report

Why "Magic number" must be positive? If it is negative I get an error: "Timestamp should be a number".

♡ 0 Reply

A **ashishhattimare** 7 months ago Report

how do we submit solution using IntelliJ. Im having trouble with it

♡ 0 Reply

CG **Christos Gogos** 9 months ago Report

animation link is missing

♡ 0 Reply

U4 **User 44396** 9 months ago Report

On check magic number there is wrong message error: "Timestamp should be a number".

♡ 0 Reply

PN **Pasha Nesterchuk** 9 months ago Fixed

Wrong answer in test #1

♡ 0 Show all