KHUSHI KANSARA **ROLL NO: 22**
SYMCA (DIVISION-B) **DATE: 30-09-2022**

PRACTICAL- 05

Aim: Implement the creation of Bitcoin Block (Genesis Block)

THEORY:

What is a node?

- A node is generally a point of intersection or connection in a telecommunications network. A node may also mean any system or physical device that is connected to a network and can execute certain functions like creating, receiving or sending information via a communication channel. The explanation of a node varies depending on the protocol layer being referred to.
- For example, a basic resident network may consist of a file server, two laptops and a fax machine. In this case, the network has four nodes, each equipped with a MAC address to uniquely identify them.
- The most popular usage of the term “node” is seen in the blockchain space. In this guide, we will explain what nodes are in more detail, including the different types of blockchain nodes being used today.
- In Ethereum, a user can run three different kinds of nodes: light, full and archive. Their differences lie in how fast they can synchronize with the entire network.
- There are many ways to run your own Ethereum node, but some popular hardware that can work on the network are DAppNode and Avado. Ethereum nodes have almost the same requirements as Bitcoin nodes, only that the former requires less computing power.
- Note that before you run an Ethereum node, it is advisable to check your bandwidth limitations first.
- Ethereum nodes are essential in keeping its blockchain network secure and reliable, as well as transparent. In fact, anyone can view the nodes and their performances on the network via Etherscan’s node tracker.
- In order to receive block rewards, you would have to run an Ethereum staking node.

Geth: Geth(Go Ethereum) is a command line interface for running Ethereum node implemented in Go Language. Using Geth you can join Ethereum network, transfer ether between accounts or even mine ethers.

Genesis Block: A genesis block refers to the first block in a blockchain and is usually hardcoded into its application’s software. A blockchain has multiple “blocks” (containing validated transactions and recorded activity data) linked together by a metaphorical chain.

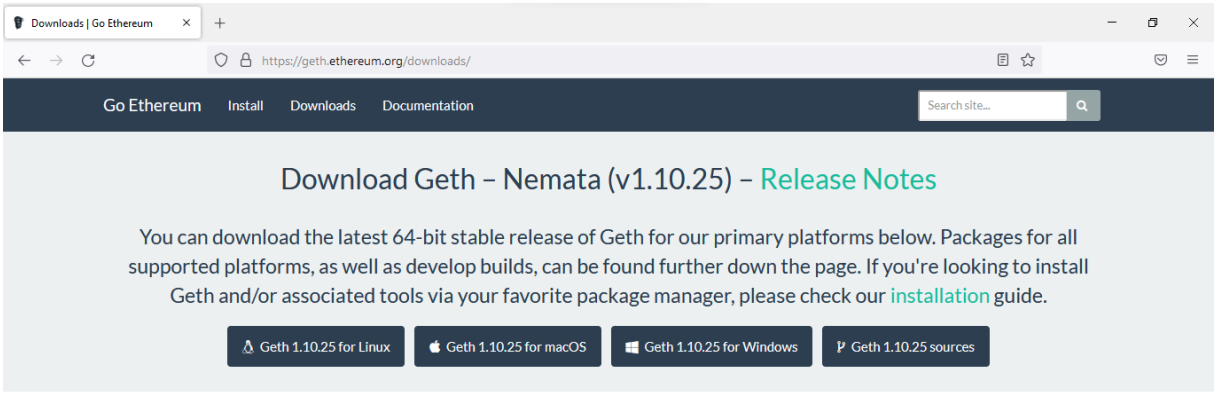
KHUSHI KANSARA
SYMCA (DIVISION-B)

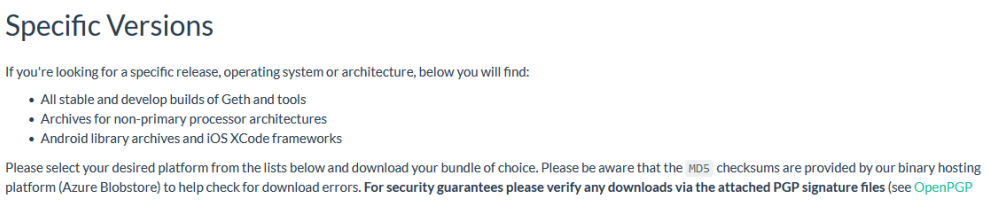
ROLL NO: 22
DATE: 30-09-2022

Each “block” of a cryptoasset contains referential data for the previous one and derives its value/legitimacy from its predecessor. The genesis block, thus, refers to the first block (Block 0 or Block 1) of a new blockchain, to which all other subsequent blocks are attached.
A genesis block is unique as it is the only block in a blockchain that does not reference a predecessor block, and in almost all cases, the first mining rewards it unlocks are unspendable.
Genesis blocks have special significance, as they form the very foundation of a blockchain and often contain interesting stories or hidden meanings. For instance, Bitcoin’s genesis block contains the now-famous message "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks" — a reference to the deteriorating financial conditions of that time and the rationale for creating cryptocurrencies like Bitcoin and Ethereum. Bitcoin’s genesis block in 2009 contained 50 BTC.
The Bitcoin genesis block is very intriguing not just for its included message, but also due to the fact that the next block was timestamped nearly six days later (the average time is 10 minutes). The present hypothesis is that the catchy Times headline enticed Satoshi Nakamoto to release Bitcoin in public, but that he actually created the genesis block earlier and altered the timestamp accordingly. After testing his software from Jan. 3, 2009, Satoshi possibly deleted all the test blocks and used the genesis block for his mainnet launch.

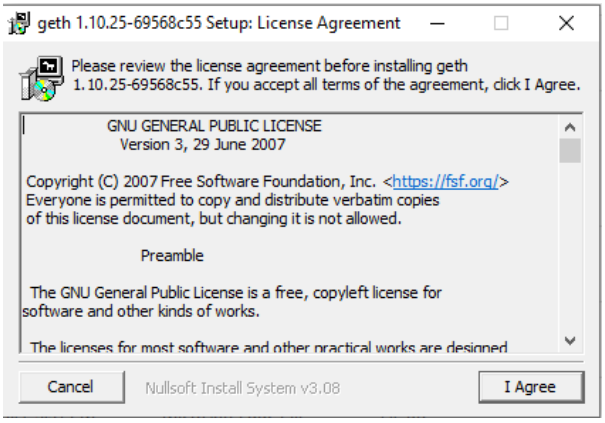
Installation of Geth:

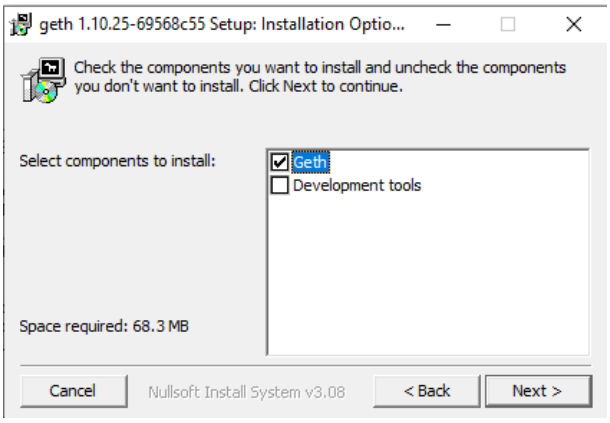
Visit here: https://geth.ethereum.org/downloads/. Download the latest release of Geth for Windows, make sure you download the 64-bit version.

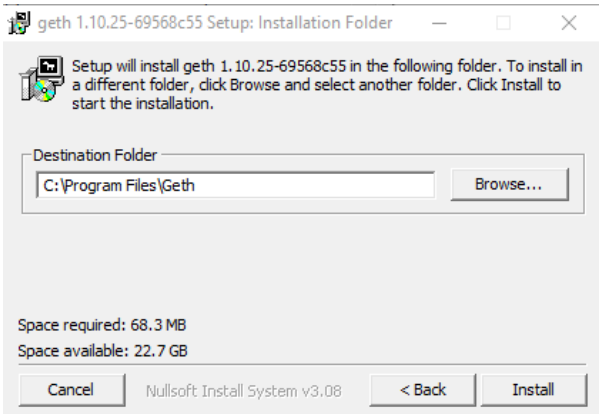


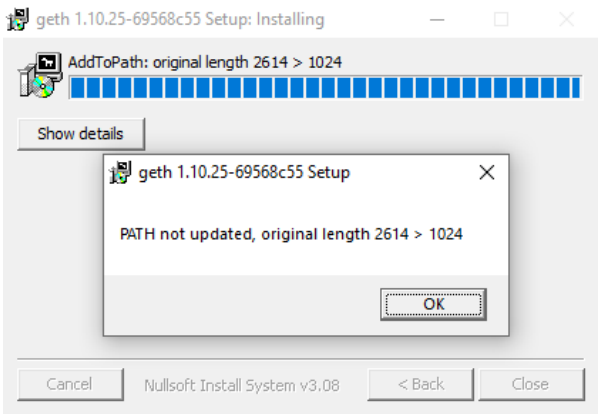


Once your download is complete, open the installer and click “I Agree” and follow the installation steps.









After the installation of Geth, set the environment variables.
After setting the environment variables, create a folder named **“PrivateChain”.**
Create a json file named **“genesis.json”** inside the folder.

genesis.json:

```
{

"config": {
"chainId": 4777,
"homesteadBlock": 0,
"eip150Block": 0,
"eip155Block": 0,
"eip158Block": 0

},
"alloc" : {},
"difficulty" : "0x400",
```

"extraData" : "",
"gasLimit" : "0x7A1200",
"parentHash" :
"0x0000000000000000000000000000000000000000000000000000000000000000",
"timestamp" : "0x00"
}

Open the command prompt from that particular location (PrivateChain) folder and execute the following commands:

1. **To initialize geth into chaindata**
   geth --datadir chaindata init genesis.json

```
D:\SYMCA_22\PrivateChain>geth --datadir PrivateChain init genesis.json
INFO [09-23|15:44:27.003] Maximum peer count                       ETH=50 LES=0 total=50
INFO [09-23|15:44:27.026] Set global gas cap                       cap=50,000,000
INFO [09-23|15:44:27.031] Allocated cache and file handles         database=D:\SYMCA_22\PrivateChain\PrivateChain\geth\c
haindata cache=16.00MiB handles=16
INFO [09-23|15:44:27.511] Opened ancient database                  database=D:\SYMCA_22\PrivateChain\PrivateChain\geth\c
haindata\ancient/chain readonly=false
INFO [09-23|15:44:27.520] Writing custom genesis block
INFO [09-23|15:44:27.522] Persisted trie from memory database      nodes=0 size=0.00B time=0s gcnodes=0 gcsize=0.00B gct
ime=0s livenodes=1 livesize=0.00B
INFO [09-23|15:44:27.528] Successfully wrote genesis state         database=chaindata hash=a685f1..43342f
INFO [09-23|15:44:27.530] Allocated cache and file handles         database=D:\SYMCA_22\PrivateChain\PrivateChain\geth\l
ightchaindata cache=16.00MiB handles=16
INFO [09-23|15:44:27.958] Opened ancient database                  database=D:\SYMCA_22\PrivateChain\PrivateChain\geth\l
ightchaindata\ancient/chain readonly=false
INFO [09-23|15:44:27.963] Writing custom genesis block
INFO [09-23|15:44:27.965] Persisted trie from memory database      nodes=0 size=0.00B time=0s gcnodes=0 gcsize=0.00B gct
ime=0s livenodes=1 livesize=0.00B
INFO [09-23|15:44:27.970] Successfully wrote genesis state         database=lightchaindata hash=a685f1..43342f

D:\SYMCA_22\PrivateChain>
```

2. **To run geth:**
   geth --datadir=./PrivateChain/

```
D:\SYMCA_22\PrivateChain>geth --datadir=./PrivateChain/
INFO [09-23|15:45:23.852] Starting Geth on Ethereum mainnet...
INFO [09-23|15:45:23.855] Bumping default cache on mainnet         provided=1024 updated=4096
INFO [09-23|15:45:23.855] Maximum peer count                       ETH=50 LES=0 total=50
WARN [09-23|15:45:23.861] Sanitizing cache to Go's GC limits       provided=4096 updated=2691
INFO [09-23|15:45:23.871] Set global gas cap                       cap=50,000,000
INFO [09-23|15:45:23.873] Allocated trie memory caches             clean=403.00MiB dirty=672.00MiB
INFO [09-23|15:45:23.881] Allocated cache and file handles         database=D:\SYMCA_22\PrivateChain\PrivateChain\geth\chaindata cache=1.31GiB handles=8192
INFO [09-23|15:45:24.058] Opened ancient database                  database=D:\SYMCA_22\PrivateChain\PrivateChain\geth\chaindata\ancient/chain readonly=false
INFO [09-23|15:45:24.065]
INFO [09-23|15:45:24.065] ---------------------------------------------------------------------------------------------------------------------------------
-----------
INFO [09-23|15:45:24.071] Chain ID:  4777 (unknown)
```

3. **IPC to interact with Geth (in another command prompt):**
   geth attach ipc:\\.\pipe\geth.ipc

```
D:\SYMCA_22\PrivateChain>geth attach ipc:\\.\pipe\geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.10.25-stable-69568c55/windows-amd64/go1.18.5
at block: 0 (Thu Jan 01 1970 05:30:00 GMT+0530 (IST))
 datadir: D:\SYMCA_22\PrivateChain\PrivateChain
 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
```

4. **Create a new account**

   > personal.newAccount()

```
D:\SYMCA_22\PrivateChain>geth attach ipc:\\.\pipe\geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/v1.10.25-stable-69568c55/windows-amd64/go1.18.5
at block: 0 (Thu Jan 01 1970 05:30:00 GMT+0530 (IST))
 datadir: D:\SYMCA_22\PrivateChain\PrivateChain
 modules: admin:1.0 debug:1.0 engine:1.0 eth:1.0 ethash:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

To exit, press ctrl-d or type exit
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0xa238bbb4d4842adf2c75271436822d7a92284cf3"
>
```

5. **Check the pre-existing accounts, coinbase account hash and the balance.**

   > eth.accounts

   > eth.coinbase

   > eth.getBalance(eth.accounts[0])

```
> eth.accounts
["0xa238bbb4d4842adf2c75271436822d7a92284cf3"]
> eth.coinbase
"0xa238bbb4d4842adf2c75271436822d7a92284cf3"
> eth.getBalance(eth.accounts[0])
0
>
```

6. **Start mining**

   > miner.start()

```
> miner.start()
null
INFO [09-23|15:56:20.516] Commit new sealing work
INFO [09-23|15:56:21.121] Successfully sealed new block
INFO [09-23|15:56:21.121] 🔨 block reached canonical chain
INFO [09-23|15:56:21.134] Commit new sealing work
INFO [09-23|15:56:21.142] 🔨 mined potential block
INFO [09-23|15:56:21.243] Commit new sealing work
INFO [09-23|15:56:22.057] Successfully sealed new block
INFO [09-23|15:56:22.058] 🔨 block reached canonical chain
INFO [09-23|15:56:22.143] Commit new sealing work
INFO [09-23|15:56:22.146] Commit new sealing work
INFO [09-23|15:56:22.149] 🔨 mined potential block
INFO [09-23|15:56:23.620] Looking for peers
INFO [09-23|15:56:23.844] Successfully sealed new block
INFO [09-23|15:56:23.845] 🔨 block reached canonical chain
```

7. **Stop mining**

   > miner.stop()

```
> miner.stop()
null
> 
```

8. **Check the accounts and the balance.**
   > eth.accounts
   > eth.getBalance(eth.accounts[0])

```
> eth.accounts
["0xa238bbb4d4842adf2c75271436822d7a92284cf3"]
> eth.getBalance(eth.accounts[0])
165000000000000000000
> 
```

9. **For the transaction, we will require 2 accounts. One would be the coinbase account and the other one would be the account which we would be transferring to.**
   > personal.newAccount()
   > eth.getBalance(eth.accounts[1])

```
> personal.newAccount()
Passphrase:
Repeat passphrase:
"0xb7824713e4543ef12ca695a7673f4929543c3d3c"
> 
> eth.getBalance(eth.accounts[1])
0
```

10. **Unlock the coinbase account and send the transaction.**
    > personal.unlockAccount(eth.accounts[0])
    > eth.sendTransaction({from: eth.coinbase, to: eth.accounts[1], value: web3.toWei(10, "ether")})

```
> personal.unlockAccount(eth.accounts[0])


Unlock account 0xa238bbb4d4842adf2c75271436822d7a92284cf3
Passphrase:
true
> eth.sendTransaction({from:eth.coinbase,to:eth.accounts[1],value:web3.toWei(10,"ether")})
"0x5721d4b7bb35e9a9fef8ee8e3dac6c51f1b6c55d263a92e96362928e0ecfbf5e"
>
> 
```

11. **In order to add the transaction in blockchain, start the mining process again.**
    > miner.start()
    > miner.stop()

```
> miner.start()
null
> miner.stop()
null
> eth.getBalance(eth.accounts[1])
10000000000000000000
>
```

12. Check the balance in terms of ether

> web3.fromWei(eth.getBalance(eth.accounts[1]), "ether")

```
> web3.fromWei(eth.getBalance(eth.accounts[1]),"ether")
10
>
```

13. Check the information about the current transaction.

>eth.getTransaction("0x7a5b57aef1bfa0bf650d2d49472de9d5072e5fa580ab7a8eb0e04b49ccef25af")

```
> eth.getTransaction("0x7a5b57aef1bfa0bf650d2d49472de9d5072e5fa580ab7a8eb0e04b49ccef25af")
{
  blockHash: "0xf15aebc8410552c7eecbba8d661db4c0d2376ad436cbcb841db19ef82b1b42e4",
  blockNumber: 72,
  chainId: "0x12a9",
  from: "0xa238bbb4d4842adf2c75271436822d7a92284cf3",
  gas: 21000,
  gasPrice: 1000000000,
  hash: "0x7a5b57aef1bfa0bf650d2d49472de9d5072e5fa580ab7a8eb0e04b49ccef25af",
  input: "0x",
  nonce: 2,
  r: "0xfabc8572b8c98ef076029471a799eae655ad241482379228820e95a68ca47f17",
  s: "0x2ad4f4a742d62974858558f60321dc97c07cdf879c7a97f6aeea7c806885ebb4",
  to: "0xb7824713e4543ef12ca695a7673f4929543c3d3c",
  transactionIndex: 1,
  type: "0x0",
  v: "0x2576",
  value: 10000000000000000000
}
>
```

14. Get information about the latest block.

> eth.getBlock("latest")

```
> eth.getBlock("latest")
{
  difficulty: 133120,
  extraData: "0xda83010a19846765746888676f312e31382e358777696e646f7773",
  gasLimit: 8854594,
  gasUsed: 0,
  hash: "0x6e67589521bd7f58a3381f24c3199f558d05c480669db806d64142ab56ee2f89",
  logsBloom: "0x000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000",
  miner: "0xa238bbb4d4842adf2c75271436822d7a92284cf3",
  mixHash: "0xc057754438263bd70ee6379cad1eed44033654d31b478e0e8a6435dcac6821da",
  nonce: "0x7a5c36e7b66c3f0f",
  number: 104,
  parentHash: "0x112e65a4686f2b3c0dd7d4f14d09e745b6ac7d1d73e36cfc49efcceb01643037",
  receiptsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  sha3Uncles: "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",
  size: 538,
  stateRoot: "0xbd359bf50efd381ec6aec5c1acc42cce5b28163a8a202db247911d0280d5b828",
  timestamp: 1663930011,
  totalDifficulty: 13743055,
  transactions: [],
  transactionsRoot: "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",
  uncles: []
}
>
```

15. Get information about a specific block.

> eth.getBlock(72)

```
> eth.getBlock(72)
{
  difficulty: 131072,
  extraData: "0xda83010a19846765746888676f312e31382e358777696e646f7773",
  gasLimit: 8582345,
  gasUsed: 42000,
  hash: "0xf15aebc8410552c7eecbba8d661db4c0d2376ad436cbcb841db19ef82b1b42e4",
  logsBloom: "0x000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000",
  miner: "0xa238bbb4d4842adf2c75271436822d7a92284cf3",
  mixHash: "0xba117b90b669e9da9fa74b7cb0541e62576a0b50304caf7a7486e3a151b897c7",
  nonce: "0x083bb1d2b61a7b0f",
  number: 72,
  parentHash: "0x9b0e9fe0f3937eb09af5d4ccb58e0f0ea266948ee19da313b790daa28c680a12",
  receiptsRoot: "0xce21669b15068c5401fcdbae4ec6a4f363e6428d8e14e0609f31351751b8301d",
  sha3Uncles: "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",
  size: 763,
  stateRoot: "0x13a13dd99ca52c28baffb8df45554264d4b01bfdd79f37c501bddfe5e1f0f60d",
  timestamp: 1663929979,
  totalDifficulty: 9514959,
  transactions: ["0x0e73eb189588b67781f0d5b0331be57d6f51338a64cc51090ae581b6dcc9aab5", "0x7a5b57aef1bfa0bf650d2d4947
  transactionsRoot: "0xd0e849d54a503d5fe9c4dc761ee05ead07adde8c2622ee1839b1ef704bffd0ba",
  uncles: []
}
>
```

Conclusion: I have gained knowledge about the creation of Bitcoin Block (Genesis Block)