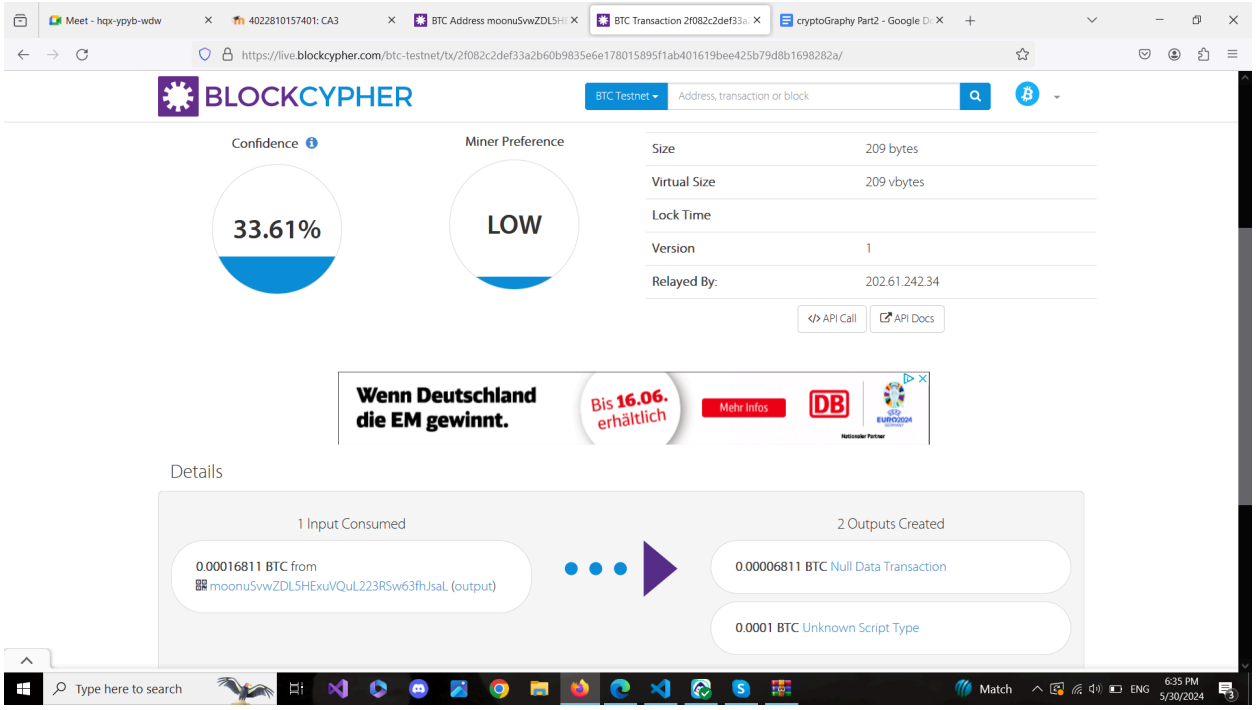
Part 1:

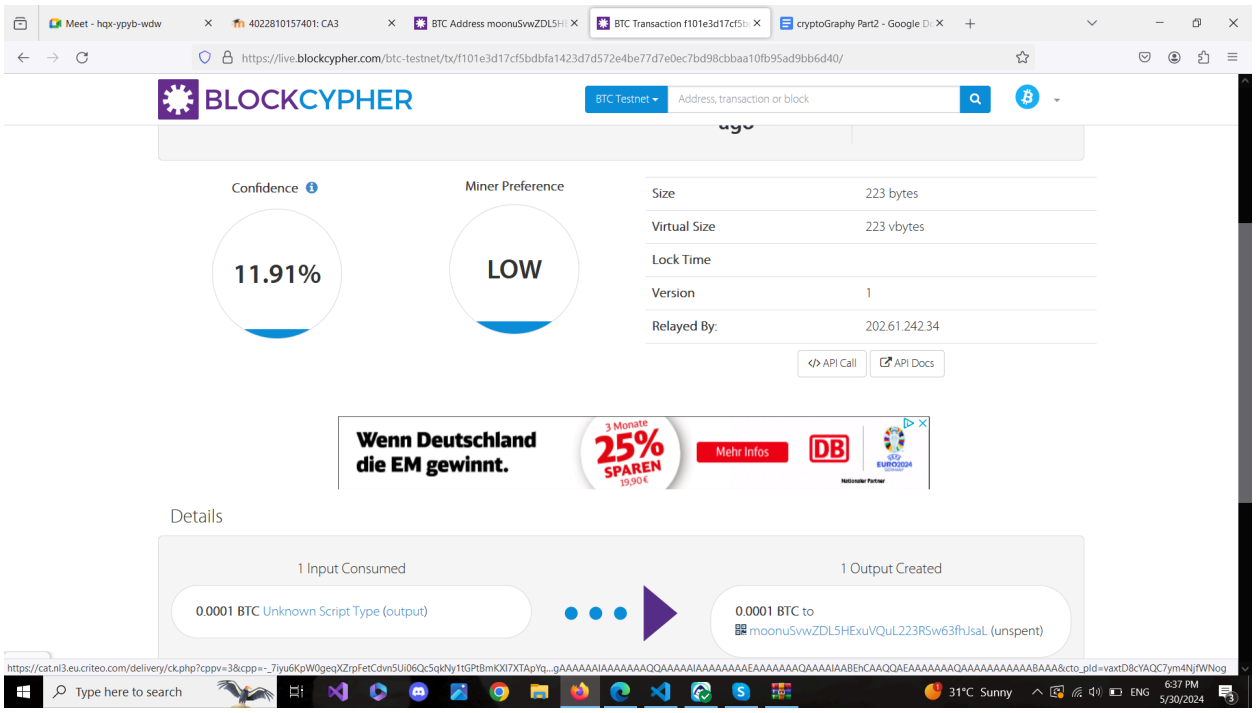
I wrote all the accounts details in the ipynb file and here we have the transaction pictures and links:

Transaction 1:

The transaction which have 2 spendable by anyone and unspendale output:
https://live.blockcypher.com/btc-testnet/tx/2f082c2def33a2b60b9835e6e178015895f1ab401619bee425b79d8b1698282a/

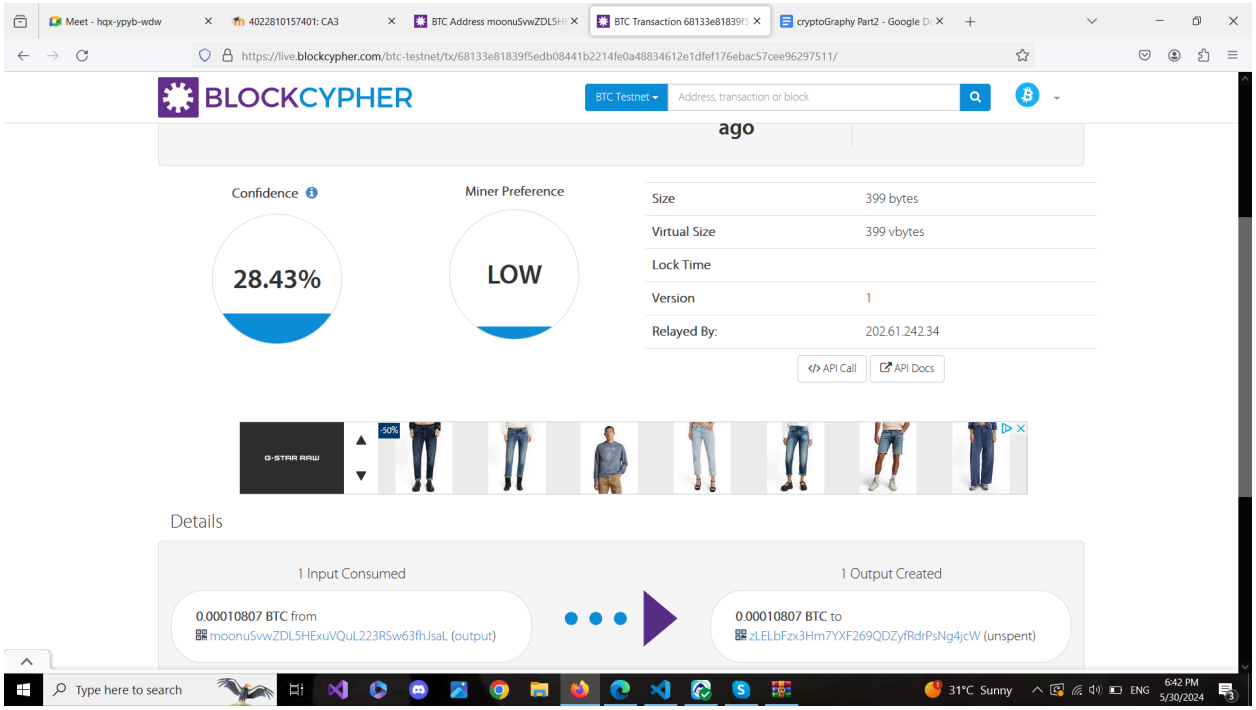


The transaction which spends the spendable by anyone index of the previous transaction: https://live.blockcypher.com/btc-testnet/tx/f101e3d17cf5bdbfa1423d7d572e4be77d7e0ec7bd98cbbaa10fb95ad9bb6d40/



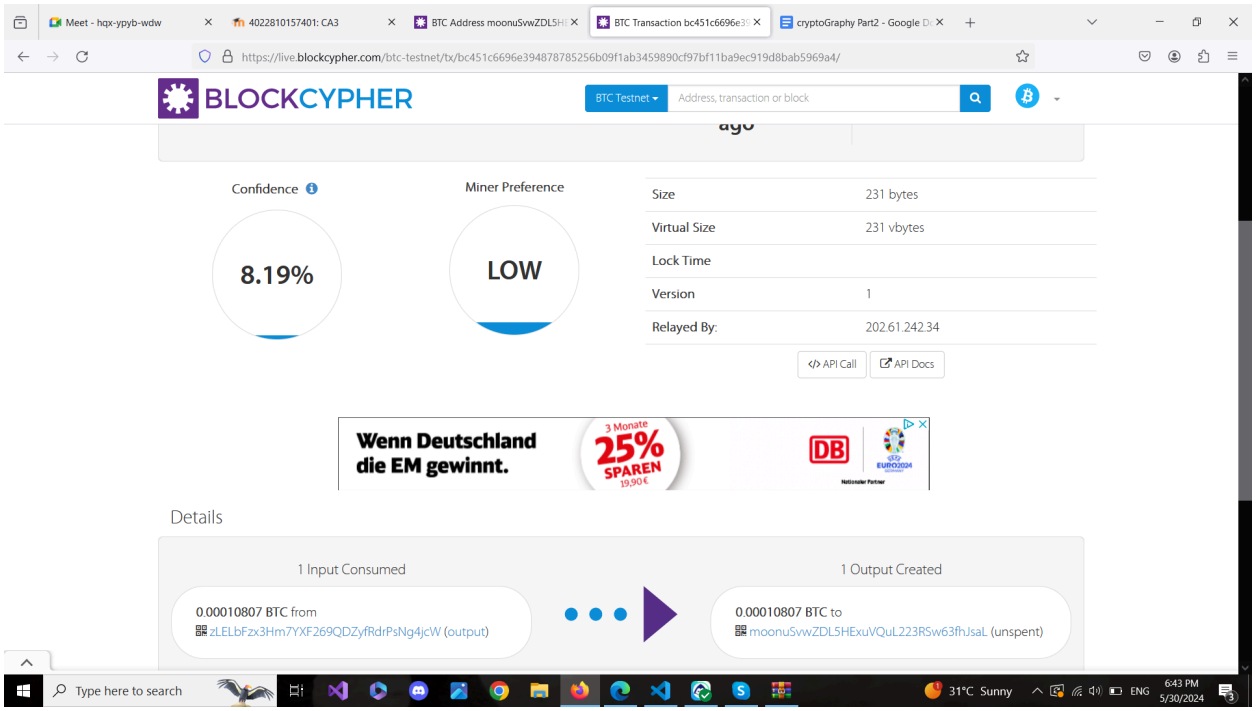
Transaction 2:

The transaction with multisig output: https://live.blockcypher.com/btc-testnet/tx/68133e81839f5edb08441b2214fe0a48834612e1dfef176ebac57cee96297511/



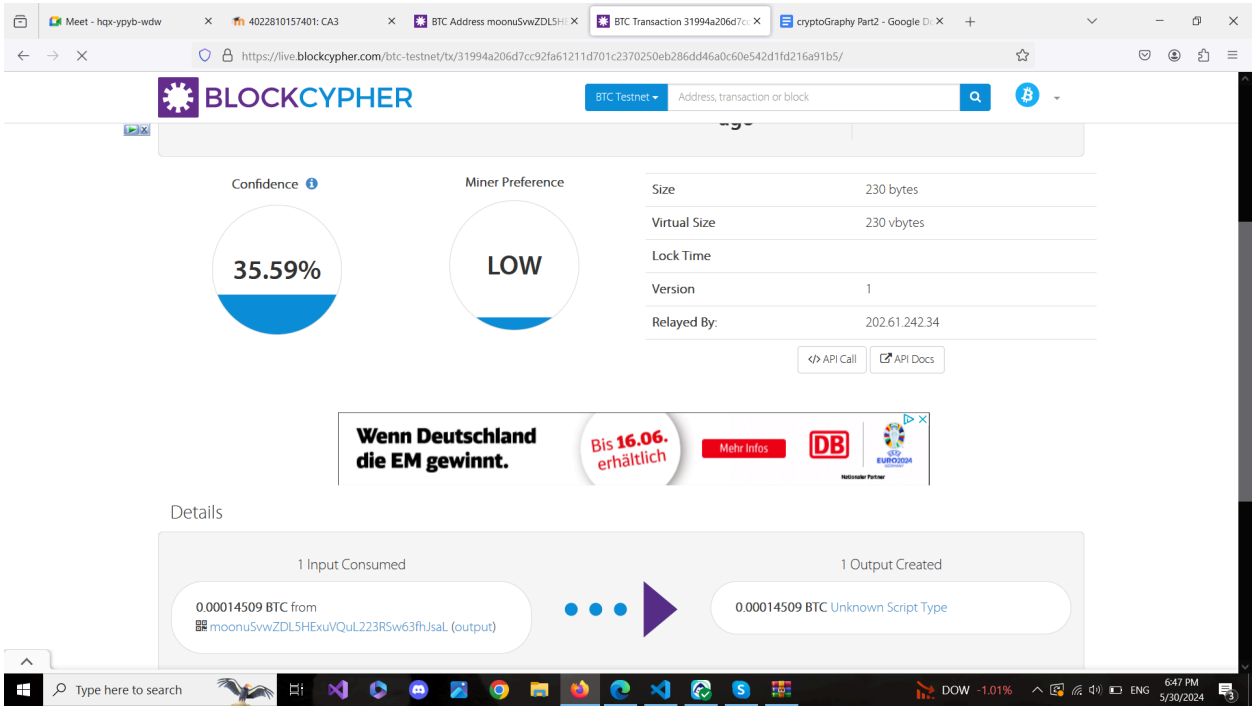
The transaction which spend the previous output:

https://live.blockcypher.com/btc-testnet/tx/bc451c6696e394878785256b09f1ab3459890cf97bf11ba9ec919d8bab5969a4/



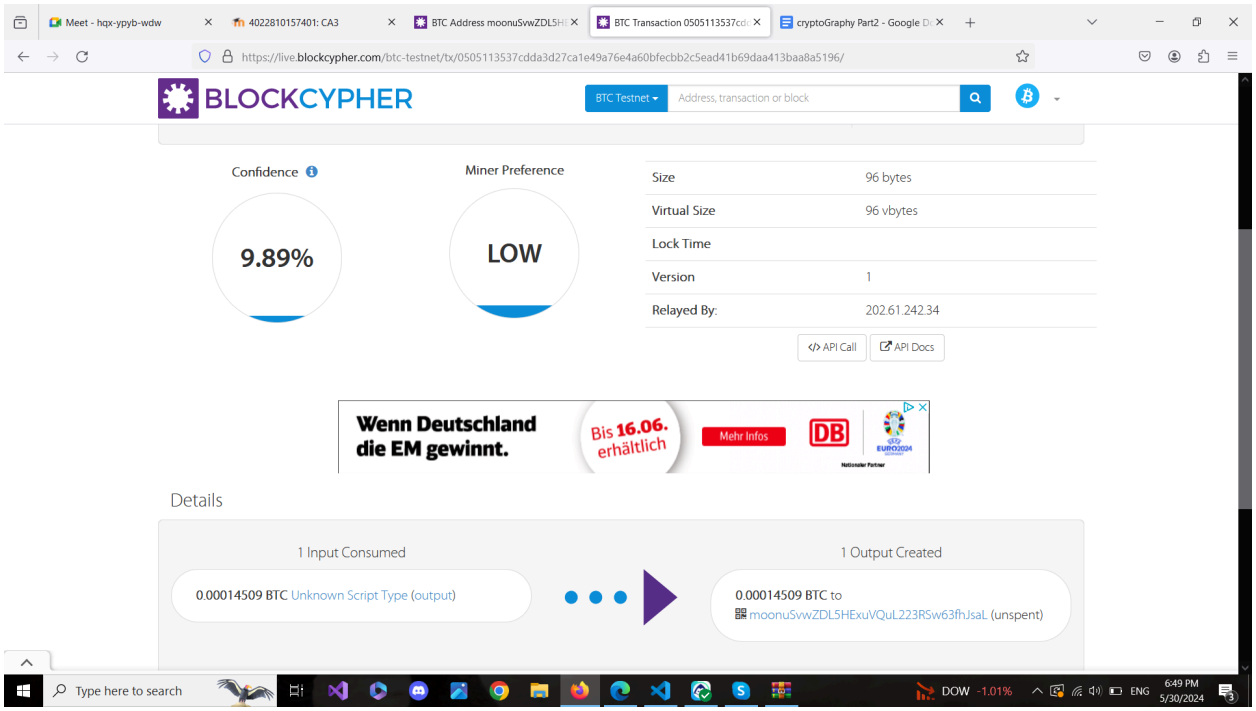
Transaction 3:

The transaction which the spender should apply the password is more than 18 years old:
https://live.blockcypher.com/btc-testnet/tx/31994a206d7cc92fa61211d701c2370250eb286dd46a0c60e542d1fd216a91b5/



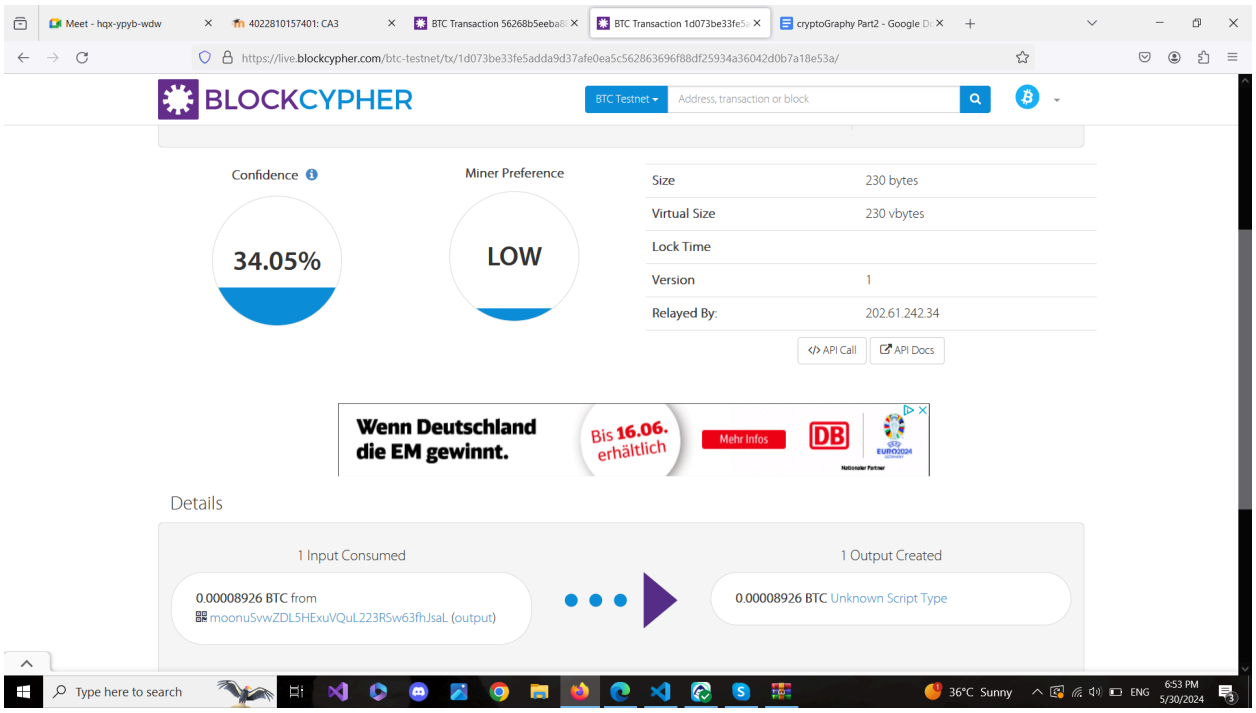
The transaction which applies the password correctly:

https://live.blockcypher.com/btc-testnet/tx/0505113537cdda3d27ca1e49a76e4a60bfecbb2c5ead41b69daa413baa8a5196/



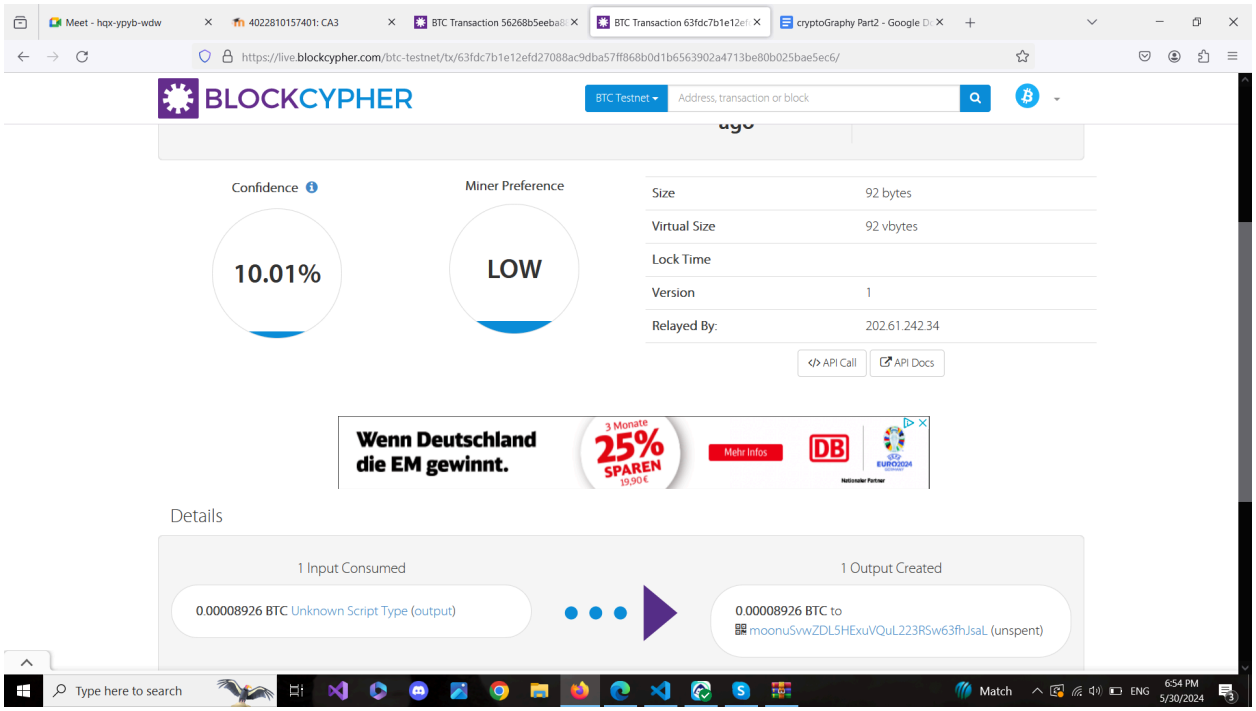
The transaction which the spender should apply the password or have age>18:

https://live.blockcypher.com/btc-testnet/tx/1d073be33fe5adda9d37afe0ea5c562863696f88df25934a36042d0b7a18e53a/



The transaction which the spender is more than 18 years old:

https://live.blockcypher.com/btc-testnet/tx/63fdc7b1e12efd27088ac9dba57ff868b0d1b6563902a4713be80b025bae5ec6/



Q1)

Reduction in Supply and Increase in Demand:

- By burning coins, the overall supply of coins in the network decreases. As supply diminishes and demand remains steady or increases, the price of the remaining coins can rise, potentially leading to value stability or even appreciation.

Financial Commitment and Incentive for Participants:

- Participants who burn their coins demonstrate a significant financial commitment. This commitment provides a strong incentive for these individuals to support and enhance the value of the network, as the

worth of their burned coins is tied to the network's stability and growth.

Preventing Centralization and Over-Control:

- Burning coins requires a financial investment, preventing excessive power concentration in the hands of a few participants. This feature can help distribute power more fairly and reduce the risk of attacks based on power centralization.

Deflationary Mechanism:

- PoB acts as a deflationary mechanism. By reducing the total number of circulating coins, the value of the remaining coins may increase, which can mitigate inflationary effects.

Q2)

No bitcoin scripts are not Turing complete because they don't contain loops.

The Pros if it was Turing complete language:

1. **Expressiveness**:
    - A Turing complete language allows for more complex and flexible smart contracts, enabling a wider range of applications and use cases.
2. **Programmability**:
    - Developers can implement more sophisticated logic and algorithms directly on the blockchain, facilitating advanced decentralized applications (dApps).
3. **Interactivity**:

- Complex interactions between contracts can be more easily handled, allowing for the creation of more intricate financial instruments and automated processes.

Part 2:

Step 1:

Here we install the geth 1.11 (because it use PoW) and run this command:

```
amirali@amirali-ROG-Strix-G513QM-G513QM:~/CryptoGraphy$ geth --help
NAME:
   geth - the go-ethereum command line interface

USAGE:
   geth [global options] command [command options] [arguments...]

VERSION:
   1.11.6-stable-ea9e62ca

COMMANDS:
   account                 Manage accounts
   attach                  Start an interactive JavaScript environment (connect to node)
   console                 Start an interactive JavaScript environment
   db                      Low level database operations
   dump                    Dump a specific block from storage
   dumpconfig              Export configuration values in a TOML format
   dumpgenesis             Dumps genesis block JSON configuration to stdout
   export                  Export blockchain into file
   export-preimages        Export the preimage database into an RLP stream
   import                  Import a blockchain file
   import-preimages        Import the preimage database from an RLP stream
   init                    Bootstrap and initialize a new genesis block
   js                      (DEPRECATED) Execute the specified JavaScript files
   license                 Display license information
   makecache               Generate ethash verification cache (for testing)
   makedag                 Generate ethash mining DAG (for testing)
   removedb                Remove blockchain and state databases
   show-deprecated-flags   Show flags that have been deprecated
   snapshot                A set of commands based on the snapshot
   verkle                  A set of experimental verkle tree management commands
   version                 Print version numbers
   version-check           Checks (online) for known Geth security vulnerabilities
   wallet                  Manage Ethereum presale wallets
   help, h                 Shows a list of commands or help for one command
```

As we can see there is a short description for each command.

Step 3:

We construct 3 nodes using command:
mkdir node01 node02 node03

Question 1)

Nodes are essential components in a cryptocurrency network, responsible for validating transactions, propagating transactions and blocks, maintaining consensus, and storing blockchain data.

Full Nodes

- **Complete Blockchain Copy**: Store the entire blockchain.
- **Transaction Validation**: Independently verify all transactions and blocks.
- **Network Support**: Relay transactions and blocks, supporting network decentralization and security.

Light Nodes (SPV Nodes)

- **Partial Blockchain Copy**: Store only block headers.
- **Relying on Full Nodes**: Verify transactions with help from full nodes, using merkle proofs.
- **Resource Efficiency**: Require less storage, computational power, and bandwidth, suitable for devices with limited resources.

Step 4:

Here we create 3 accounts:

Account 1:

```
amirali@amirali-ROG-Strix-G513QM-G513QM:~/CryptoGraphy$ geth --datadir "node01" account new
INFO [05-28|11:13:59.305] Maximum peer count                       ETH=50 LES=0 total=50
INFO [05-28|11:13:59.305] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no such file or directory"
Your new account is locked with a password. Please give a password. Do not forget this password.
Password:
Repeat password:

Your new key was generated

Public address of the key:   0x82dBEfF6e3c87b429A5e73a8401B121a2d7fCce6
Path of the secret key file: node01/keystore/UTC--2024-05-28T15-14-02.885673310Z--82dbeff6e3c87b429a5e73a8401b121a2d7fcce6

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your funds!
- You must BACKUP your key file! Without the key, it's impossible to access account funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt the key!
```

Account 2:

```
amirali@amirali-ROG-Strix-G513QM-G513QM:~/CryptoGraphy$ geth --datadir "node02" account new
INFO [05-28|11:23:14.991] Maximum peer count                       ETH=50 LES=0 total=50
INFO [05-28|11:23:14.991] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no such file or directory"
Your new account is locked with a password. Please give a password. Do not forget this password.
Password:
Repeat password:

Your new key was generated

Public address of the key:   0xcE4AF4889e5f154303fEfb05CefC147C5290F511
Path of the secret key file: node02/keystore/UTC--2024-05-28T15-23-22.509176830Z--ce4af4889e5f154303fefb05cefc147c5290f511

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your funds!
- You must BACKUP your key file! Without the key, it's impossible to access account funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt the key!
```

Account 3:

```
amirali@amirali-ROG-Strix-G513QM-G513QM:~/CryptoGraphy$ geth --datadir "node03" account new
INFO [05-28|11:23:57.925] Maximum peer count                       ETH=50 LES=0 total=50
INFO [05-28|11:23:57.925] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no such file or directory"
Your new account is locked with a password. Please give a password. Do not forget this password.
Password:
Repeat password:

Your new key was generated

Public address of the key:   0x1d5634c61A5c5d336819f1A301dc6e6D235dC36C
Path of the secret key file: node03/keystore/UTC--2024-05-28T15-24-02.321010291Z--1d5634c61a5c5d336819f1a301dc6e6d235dc36c

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your funds!
- You must BACKUP your key file! Without the key, it's impossible to access account funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt the key!
```

Here we can perform step 2, because now we have the addresses we wanted. So we fill the genesis file with this informations.

Step 5:

Here we init our nodes.
Node 1:

```
amirali@amirali-ROG-Strix-G513QM-G513QM:~/CryptoGraphy$ geth --datadir "node01" init genesis.json
INFO [05-28|13:08:20.450] Maximum peer count                       ETH=50 LES=0 total=50
INFO [05-28|13:08:20.450] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no such file or directory"
INFO [05-28|13:08:20.455] Set global gas cap                       cap=50,000,000
INFO [05-28|13:08:20.458] Using leveldb as the backing database
INFO [05-28|13:08:20.458] Allocated cache and file handles         database=/home/amirali/CryptoGraphy/node01/geth/chaindata cache=16.00MiB handles=16
INFO [05-28|13:08:20.469] Using LevelDB as the backing database
INFO [05-28|13:08:20.473] Opened ancient database                  database=/home/amirali/CryptoGraphy/node01/geth/chaindata/ancient/chain readonly=false
INFO [05-28|13:08:20.473] Writing custom genesis block
INFO [05-28|13:08:20.475] Persisted trie from memory database      nodes=4 size=585.00B time="629.43µs" gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [05-28|13:08:20.476] Successfully wrote genesis state         database=chaindata hash=35e255..864007
INFO [05-28|13:08:20.477] Using leveldb as the backing database
INFO [05-28|13:08:20.477] Allocated cache and file handles         database=/home/amirali/CryptoGraphy/node01/geth/lightchaindata cache=16.00MiB handles=16
INFO [05-28|13:08:20.482] Using LevelDB as the backing database
INFO [05-28|13:08:20.493] Opened ancient database                  database=/home/amirali/CryptoGraphy/node01/geth/lightchaindata/ancient/chain readonly=false
INFO [05-28|13:08:20.493] Writing custom genesis block
INFO [05-28|13:08:20.494] Persisted trie from memory database      nodes=4 size=585.00B time="286.708µs" gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [05-28|13:08:20.495] Successfully wrote genesis state         database=lightchaindata hash=35e255..864007
```

Node 2:

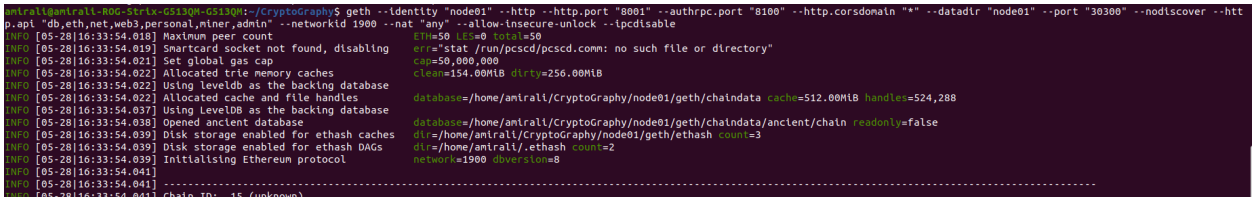
```
amirali@amirali-ROG-Strix-G513QM-G513QM:~/CryptoGraphy$ geth --datadir "node02" init genesis.json
INFO [05-28|13:09:32.014] Maximum peer count                       ETH=50 LES=0 total=50
INFO [05-28|13:09:32.015] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no such file or directory"
INFO [05-28|13:09:32.017] Set global gas cap                       cap=50,000,000
INFO [05-28|13:09:32.018] Using leveldb as the backing database
INFO [05-28|13:09:32.018] Allocated cache and file handles         database=/home/amirali/CryptoGraphy/node02/geth/chaindata cache=16.00MiB handles=16
INFO [05-28|13:09:32.021] Using LevelDB as the backing database
INFO [05-28|13:09:32.028] Opened ancient database                  database=/home/amirali/CryptoGraphy/node02/geth/chaindata/ancient/chain readonly=false
INFO [05-28|13:09:32.028] Writing custom genesis block
INFO [05-28|13:09:32.029] Persisted trie from memory database      nodes=4 size=585.00B time="118.174µs" gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [05-28|13:09:32.030] Successfully wrote genesis state         database=chaindata hash=35e255..864007
INFO [05-28|13:09:32.031] Using leveldb as the backing database
INFO [05-28|13:09:32.031] Allocated cache and file handles         database=/home/amirali/CryptoGraphy/node02/geth/lightchaindata cache=16.00MiB handles=16
INFO [05-28|13:09:32.035] Using LevelDB as the backing database
INFO [05-28|13:09:32.047] Opened ancient database                  database=/home/amirali/CryptoGraphy/node02/geth/lightchaindata/ancient/chain readonly=false
INFO [05-28|13:09:32.047] Writing custom genesis block
INFO [05-28|13:09:32.048] Persisted trie from memory database      nodes=4 size=585.00B time="102.46µs"  gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [05-28|13:09:32.049] Successfully wrote genesis state         database=lightchaindata hash=35e255..864007
```

Node 3:

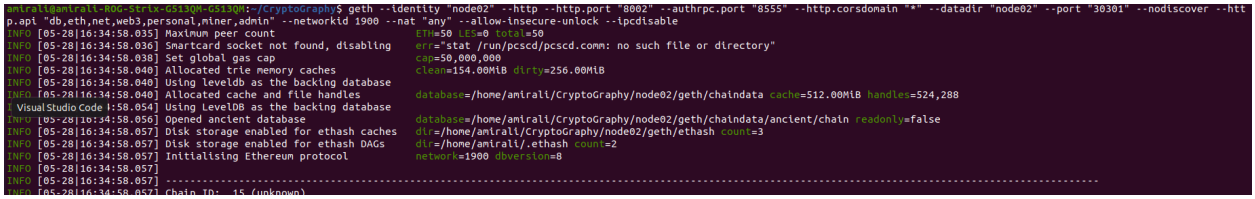
```
amirali@amirali-ROG-Strix-G513QM-G513QM:~/CryptoGraphy$ geth --datadir "node03" init genesis.json
INFO [05-28|13:10:55.977] Maximum peer count                       ETH=50 LES=0 total=50
INFO [05-28|13:10:55.977] Smartcard socket not found, disabling    err="stat /run/pcscd/pcscd.comm: no such file or directory"
INFO [05-28|13:10:55.979] Set global gas cap                       cap=50,000,000
INFO [05-28|13:10:55.980] Using leveldb as the backing database
INFO [05-28|13:10:55.980] Allocated cache and file handles         database=/home/amirali/CryptoGraphy/node03/geth/chaindata cache=16.00MiB handles=16
INFO [05-28|13:10:55.984] Using LevelDB as the backing database
INFO [05-28|13:10:55.996] Opened ancient database                  database=/home/amirali/CryptoGraphy/node03/geth/chaindata/ancient/chain readonly=false
INFO [05-28|13:10:55.996] Writing custom genesis block
INFO [05-28|13:10:55.996] Persisted trie from memory database      nodes=4 size=585.00B time="39.043µs" gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [05-28|13:10:55.997] Successfully wrote genesis state         database=chaindata hash=35e255..864007
INFO [05-28|13:10:55.998] Using leveldb as the backing database
INFO [05-28|13:10:55.998] Allocated cache and file handles         database=/home/amirali/CryptoGraphy/node03/geth/lightchaindata cache=16.00MiB handles=16
INFO [05-28|13:10:56.003] Using LevelDB as the backing database
INFO [05-28|13:10:56.014] Opened ancient database                  database=/home/amirali/CryptoGraphy/node03/geth/lightchaindata/ancient/chain readonly=false
INFO [05-28|13:10:56.014] Writing custom genesis block
INFO [05-28|13:10:56.015] Persisted trie from memory database      nodes=4 size=585.00B time="70.262µs" gcnodes=0 gcsize=0.00B gctime=0s livenodes=1 livesize=0.00B
INFO [05-28|13:10:56.016] Successfully wrote genesis state         database=lightchaindata hash=35e255..864007
```

And Start Them:

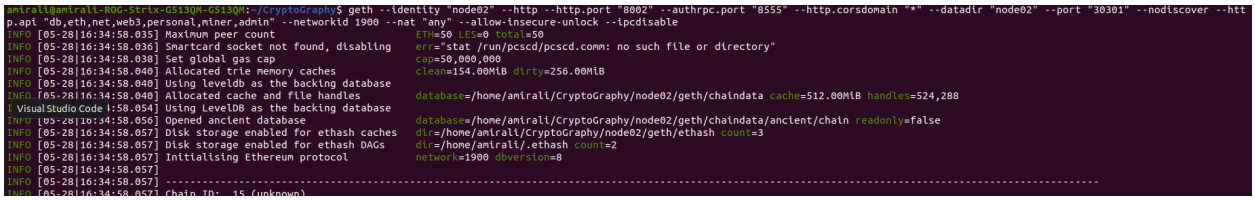
Node 1:



Node 2:



Node 3:



Step 6:

Now we connect to all nodes using 3 different clients and see the node info with admin.nodeInfo command:

Node 1:

Node 2:

Node 3:

```
> amirali@amirali-ROG-Strix-G513QM-G513QM:~$ geth attach http://127.0.0.1:8002
WARN [05-28|16:47:19.354] Enabling deprecated personal namespace
Welcome to the Geth JavaScript console!

instance: Geth/node02/v1.11.6-stable-ea9e62ca/linux-amd64/go1.20.3
at block: 0 (Wed Dec 31 1969 19:00:00 GMT-0500 (EST))
 datadir: /home/amirali/CryptoGraphy/node02
 modules: admin:1.0 eth:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 web3:1.0

To exit, press ctrl-d or type exit
> admin.nodeInfo
{
  enode: "enode://0dc06b5003352a781bc8c37ba1c5cb60b6b18177dee5acb7318533faf317be26bc55fef05e2644dd761dd8cb06d123cc322299484a85060c3de277293adea821@192.168.0.118:30301?discport=0",
  enr: "enr:-Jy4QBJcOdwJJf9IMSKbS-W4taODIWGDZ0TagNSLG2Y617JfSuNKaD1HfgOx7l7f_P97hJu7I66vqcpYu9NBWudu_QqGAY_A6cmWg2V0aMfGhK2OgSuAgmlkgnY0gmlwhMCoAHaJc2VjcDI1NmsxoQMNwGtQAzUqeBvIw3uhxctgtrGBd97lrLcxhTP68xe-JoRzbmFwwIN0Y3CCdl0",
  id: "6cbf9c9b9fe9b17ae29252d4cb28e877c6d019ba69583ef37aee798b225ccd16",
  ip: "192.168.0.118",
  listenAddr: "[::]:30301",
  name: "Geth/node02/v1.11.6-stable-ea9e62ca/linux-amd64/go1.20.3",
  ports: {
    discovery: 0,
    listener: 30301
  },
  protocols: {
    eth: {
      config: {
        chainId: 15,
        eip150Block: 0,
        eip155Block: 0,
        eip158Block: 0,
        homesteadBlock: 0
      },
      difficulty: 400000,
      genesis: "0x35e255a04728e19e4cb9bd5f6227543fedac3270e79ff03c8a121ede2d864007",
      head: "0x35e255a04728e19e4cb9bd5f6227543fedac3270e79ff03c8a121ede2d864007",
      network: 1900
    },
    snap: {}
  }
}
```

Step 7:

Here we want to connect the nodes together, here we use client connected to node 1 and peer it to other nodes:

Run addPeer command to connect node 1 to node 2 and 3 and finally printing peer count for node 1:

```
> admin.addPeer("enode://0dc06b5003352a781bc8c37ba1c5cb60b6b18177dee5acb7318533faf317be26bc55fef05e2644dd761dd8cb06d123cc322299484a85060c3de277293adea821@192.168.0.118:30301?discport=0")
true
> admin.addPeer("enode://f9d3c857d1c00f86065b9b95d514bee722778ba83eef746436dffd5fbd128a448377d09fc4e1978667735db8e8973c874ba4f6e7bcebfa54e917b28dbd779273@192.168.0.118:30302?discport=0")
true
> net.peerCount
2
>
```

Node 2 peer Count:

```
> net.peerCount
1
```

Node 3 peer count:

```
> net.peerCount
1
```

Step 8:

Here we check each node account address and ballance:
Node1:

```
> eth.accounts
["0x82dbeff6e3c87b429a5e73a8401b121a2d7fcce6"]
> eth.getBalance(eth.accounts[0])
10000000000810100146
>
```

Node 2:

```
> eth.accounts
["0xce4af4889e5f154303fefb05cefc147c5290f511"]
> eth.getBalance(eth.accounts[0])
20000000000810100146
>
```

Node 3:

```
> eth.accounts
["0x1d5634c61a5c5d336819f1a301dc6e6d235dc36c"]
> eth.getBalance(eth.accounts[0])
15000000000810100146
>
```

We want to send 10k coin from node 1 to node 2 so we unlock account first and then create the transaction:

```
> personal.unlockAccount(eth.accounts[0])
Unlock account 0x82dbeff6e3c87b429a5e73a8401b121a2d7fcce6
Passphrase:
true
> eth.sendTransaction({from:eth.accounts[0], to:"0xce4af4889e5f154303fefb05cefc147c5290f511", value:10000})
"0xf475042292bb6d040c0551cb7263032920876a9f61b86ca598ff927563cb5737"
> eth.getBalance(eth.accounts[0])
10000000000810100146
```

As we can see the transaction has been created but it's not still in any block so the balance didn't change.

Step 9:

Here we start mining on node 1:

```
> miner.setEtherbase(eth.accounts[0])
true
> miner.start()
null
> miner.stop()
null
> eth.getBalance(eth.accounts[0])
626000000000810090146
```

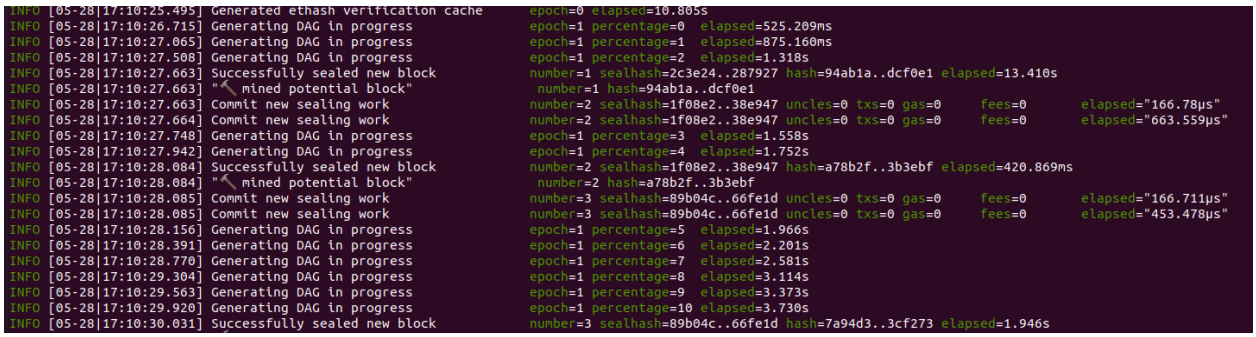
As we can see the balance of this node has decreased 10k because of the acceptance of the transaction and it has increased so much because of the block reward achieved.

And here is the balance of account 2 which has increased by 10k:

```
> eth.getBalance(eth.accounts[0])
2000000000810110146
>
```

Here are the mining logs:

```
INFO [05-28|17:10:23.080] Generating DAG in progress                epoch=0 percentage=78 elapsed=8.390s
INFO [05-28|17:10:23.183] Generating DAG in progress                epoch=0 percentage=79 elapsed=8.493s
INFO [05-28|17:10:23.286] Generating DAG in progress                epoch=0 percentage=80 elapsed=8.596s
INFO [05-28|17:10:23.389] Generating DAG in progress                epoch=0 percentage=81 elapsed=8.699s
INFO [05-28|17:10:23.508] Generating DAG in progress                epoch=0 percentage=82 elapsed=8.819s
INFO [05-28|17:10:23.648] Generating DAG in progress                epoch=0 percentage=83 elapsed=8.958s
INFO [05-28|17:10:23.786] Generating DAG in progress                epoch=0 percentage=84 elapsed=9.096s
INFO [05-28|17:10:23.906] Generating DAG in progress                epoch=0 percentage=85 elapsed=9.217s
INFO [05-28|17:10:24.010] Generating DAG in progress                epoch=0 percentage=86 elapsed=9.320s
INFO [05-28|17:10:24.114] Generating DAG in progress                epoch=0 percentage=87 elapsed=9.424s
INFO [05-28|17:10:24.218] Generating DAG in progress                epoch=0 percentage=88 elapsed=9.528s
INFO [05-28|17:10:24.321] Generating DAG in progress                epoch=0 percentage=89 elapsed=9.631s
INFO [05-28|17:10:24.425] Generating DAG in progress                epoch=0 percentage=90 elapsed=9.735s
INFO [05-28|17:10:24.528] Generating DAG in progress                epoch=0 percentage=91 elapsed=9.838s
INFO [05-28|17:10:24.632] Generating DAG in progress                epoch=0 percentage=92 elapsed=9.943s
INFO [05-28|17:10:24.736] Generating DAG in progress                epoch=0 percentage=93 elapsed=10.046s
INFO [05-28|17:10:24.839] Generating DAG in progress                epoch=0 percentage=94 elapsed=10.150s
INFO [05-28|17:10:24.943] Generating DAG in progress                epoch=0 percentage=95 elapsed=10.253s
INFO [05-28|17:10:25.047] Generating DAG in progress                epoch=0 percentage=96 elapsed=10.357s
INFO [05-28|17:10:25.152] Generating DAG in progress                epoch=0 percentage=97 elapsed=10.462s
INFO [05-28|17:10:25.259] Generating DAG in progress                epoch=0 percentage=98 elapsed=10.569s
INFO [05-28|17:10:25.493] Generating DAG in progress                epoch=0 percentage=99 elapsed=10.803s
```



Question 2)

Generating DAG in Progress:

- **Description**: This message indicates that the process of generating the Directed Acyclic Graph (DAG) is underway.
- **Details**: In Ethereum mining, the DAG is a large dataset used in the Ethash proof-of-work algorithm. It is regenerated every epoch (approximately every 30,000 blocks) and must be loaded into the memory of the mining hardware.

Generated Ethash Verification Cache:

- **Description**: This output signifies that the Ethash verification cache has been created.
- **Details**: The verification cache is a smaller, more efficient structure derived from the DAG that allows miners to quickly verify the validity of potential solutions. This helps speed up the mining process.

Successfully Sealed New Block:

- **Description**: This message indicates that a new block has been successfully created (sealed).
- **Details**: Sealing a block means the miner has found a valid nonce that, when combined with the block's other data, produces a hash below the network's target difficulty. This block is then ready to be added to the blockchain.

Mined Potential Block:

- **Description**: This indicates that a block has been mined and is a candidate for inclusion in the blockchain.
- **Details**: A potential block is one that has been successfully mined but has not yet been confirmed by the network. It awaits validation and inclusion by other nodes.

Commit New Swaling Work:

- **Description**: This message suggests that the miner is committing to new mining work.
- **Details**: Swaling work refers to the process of starting new mining tasks. After a block is mined, the miner begins working on the next block, constantly updating and committing to the latest work.

Question 3:

No we simply can't do that, here are the reasons why:

Insufficient Hash Power:

- To mine 19,722,000 blocks quickly, you would need an astronomical amount of hash power, far exceeding the combined power of the entire Ethereum network. Even if you had the resources, mining that many blocks faster than the rest of the network is practically impossible due to the sheer computational power required.

Difficulty Adjustment:

- Ethereum's PoW algorithm adjusts the mining difficulty to ensure a stable block time. As you try to mine more blocks quickly, the difficulty would increase exponentially, making it increasingly harder to mine subsequent blocks at a fast pace.

Economic Constraints:

- Attempting such an attack would be prohibitively expensive. The electricity and hardware costs to generate such a high volume of blocks would be enormous, and there would be no guarantee of success.