upGrad

Interview Questions:
Smart Contract Development on Ethereum

Q1	Write the geth command to start an Ethereum node client with an Remote Procedure Call (RPC) connection. Explain the various flags that are used and the reason to add them.
Ans	```geth --data dir <path to your testnet directory> --networkid 12345 --rpc --rpcaddr "localhost" --rpcport 8545 --rpccorsdomain "*" --rpcapi "eth,net,web3,personal,admin,mine" --mine --minerthreads 1 --nodiscover --maxpeers 0 miner``` • datadir: It specifies the path of the directory which will store your chain data. This should be the same folder as specified in the earlier command. • networkid: It should be the same network id as specified earlier. • rpcapi “mine”: It enables running miner commands in JavaScript using Ethereum web API like web3.js or Ethereum JS. • mine: It enables mining. • minerthreads: It is the number of CPU threads which are to be used for mining.

Q2	Name the network ids for Ethereum Mainnet, Ropsten Testnet, Rinkeby Testnet and Kovan Testnet?
Ans	a. Mainnet: 1 b. Ropsten: 3 c. Rinkeby: 4 d. Kovan: 42

Q3	How many peers can be added to an Ethereum node at a time?
Reference	https://github.com/ethereum/go-ethereum/wiki/command-line-options
Ans	A maximum of 50 peers can be added to an Ethereum node at a time.

Q4	What is a txpool?
Ans	Txpool stands for the transaction pool of the Ethereum network. A txpool is a storage where all unconfirmed transactions get stored.

Q5	What is the difference between pending and queued transactions in txpool?
Reference	https://ethereum.stackexchange.com/questions/9137/what-is-the-difference-between-a-pending-transaction-and-a-queued-transaction-in
Ans	Pending transactions are defined as all the transactions which are ready to be processed and added in a block. Queued transactions' nonce is not in a sequence, and hence they are yet to be queued in the txpool.

upGrad

Q6	What is the gas limit of any transaction?
Ans	Gas limit is the value which determines the maximum gas that can be spent in a transaction.

Q7	How to set up a private Ethereum network to accept 0 gas price?
Reference	https://stackoverflow.com/questions/49318479/how-to-make-sure-transactions-take-0-fee-in-a-private-ethereum-blockchain
Ans	Miners can be controlled in a private network and, hence, you can only accept transactions which have 0 gas price.

Q8	What are the advantages of the Clique protocol?
Ans	The Clique protocol is also called PoA or the proof of authority consensus algorithm. Its advantages are: - Faster transaction - Low computation requirement - Block creation control group

Q9	What are the parameters which determine the TPS in a private Ethereum network?
Reference	https://publik.tuwien.ac.at/files/publik_280601.pdf
Ans	In Ethereum, TPS or transactions per second depends on multiple parameters as mentioned in the table:

Parameter	Description
Block frequency	Time between two succeeding blocks
Block size	Amount of transactions fitting in a block
Workload type	Smart contract
Node configuration	CPU, RAM, network speed
Network size	Number of nodes
Network structure	Structure of the blockchain network
Workload quantity	Amount of transactions to be processed
Amount of miners/sealers	Actively participating nodes
Blockchain client and API	E.g. Geth or Parity, web3.js or web3.py

Q10	What is the difference between boot node and RPC node?
Ans	The bootnode does not have data records, but the RPC node has an exact copy of the blockchain.

Q11	What is Inter Process Communication (IPC)?
Ans	IPC or inter process communications is a protocol that generally works on your local computers. It uses a computer local file system.

Q12	Which process can be used to submit faster transactions – IPC or RPC?
Ans	IPC does not have network latency. So, it works well when submitting transactions.

Q13	How to get events from a private blockchain?
Ans	Events from a private Ethereum blockchain requires WS (WebSocket).

Q14	What is a truffle migration file?
Ans	Migrations are JavaScript files that help you deploy contracts to the Ethereum network. These files are responsible for staging your deployment tasks, and are written under the assumption that your deployment needs will change over time. As your project evolves, you will create new migration scripts to further this evolution on the blockchain. A history of previously run migrations is recorded on-chain.

Q15	How to create CI/CD pipeline for Ethereum smart contract deployment?
Ans	Truffle can be configured to set a complete CI/CD pipeline for Ethereum smart contracts.

Q16	What is the function of each of the following keywords in Solidity? 1. Pragma 2. Return 3. Contract 4. This
Ans	- Pragma: The pragma keyword can be used to enable certain compiler features or checks. A pragma directive is always local to a source file. So, you have to add the pragma to all your files if you want to enable it in all of your projects. If you import another file, the pragma from that file will not automatically apply to the importing file. - Return: The return keyword is used to exit a function. - Contract: The contract keyword is used to start a new contract and is similar to how the class keyword is used in Java. - This: The this keyword refers to the current contract, explicitly convertible to address using address (this).

Q17	How many output parameters are supported in a function in Solidity?

Ans	There is no upper limit for the output parameters.

Q18	Write a sample modifier 'myModifier' which takes in a number as an input parameter and terminates the function it 'modifies' if the number is even.
Ans	modifier myModifier(uint256 a) { require(a%2==0); _; }

Q19	Write a contract to model an automobile registry system. The contract should have the following components: The details of the contract are as follows. Design a simple contract accordingly. a. A struct to store three automobile related details: i. Registration number ii. Insurance number iii. PUC certificate number b. A mapping between the automobile details and the automobile owner address c. A second mapping between the address and the owner details such as name, age and gender d. A constructor to initialise two individuals – you and your best friend and your automobile ownership e. A function 'swapOwnership' which takes in the two addresses and swaps their automobile ownership f. A modifier 'onlyRTO' which makes sure that the function 'swapOwnership' is called only if the RTO address (assume it to be 0x123456789) calls the function. g. An event 'showDetails' that prints the automobile details as logs h. A function 'show' which takes in an address as the input parameter and prints the automobile details mapped to that address as logs

Q20	What are event topics? What keyword is used to declare an input parameter for an event as an event topic? How many event topics are allowed in an event in Solidity?
Ans	The explanation can be as follows: a. EVM uses low-level primitives called logs to map them to high-level Solidity constructs called events. Logs may contain different topics that are indexed arguments. Consider an event: `emit PersonCreated(uint indexed age, uint indexed height);` And you fire it in MyContract: `function foobar() {` `  PersonCreated(26, 176);` `}` This will create a low-level EVM log entry with topics: - 0x6be15e8568869b1e100750dd5079151b32637268ec08d199b318b793181b8a7d (Keccak-256 hash of PersonCreated(uint256,uint256)) - 0x36383cc9cfbf1dc87c78c2529ae2fcd4e3fc4e575e154b357ae3a8b2739113cf (Keccak-256 hash of age), value 26 - 0x048dd4d5794e69cea63353d940276ad61f89c65942226a2bb5bd352536892f82 (Keccak-256 hash of height), value 176 Internally, your Ethereum node (Geth/Parity) will index arguments to build on indexable search indexes, so that you can easily do lookups by value later. As creating indexes takes additional disk space, indexed parameters in events have additional gas cost. However, indexes are required to look up any meaningful scale of events by value later. Now, in the web3 client, you would want to watch creation events of all persons who are of age 26. For this, you can simply do the following: `var createdEvent = myContract.PersonCreated({age: 26});` `createdEvent.watch(function(err, result) {` `  if (err) {` `  console.log(err)`

	return; } console.log("Found ", result); }) b) 3 indexed parameters

Q21	Name the four visibilities supported by functions and state variables in Solidity.
Ans	The four visibilities supported by Solidity are public, private, internal and external.

Q22	What is the default visibility of a state variable in Solidity?
Ans	The default visibility of a state variable in Solidity is private.

Q23	What is the SafeMath library used for in Ethereum smart contract?
Reference	https://ethereumdev.io/safemath-protect-overflows/
Ans	While working with numbers in Solidity, there are high chances of overflows or underflows. For example, if a number is declared as uint, it has a limit of 0 to 2^256. Any number above this will result in an overflow. To prevent these types of overflows and underflows, the SafeMath library is used in Solidity. It will check overflows in your contract.

Q24	What do you mean by gas price? Who sets the gas price for an Ethereum transaction?
Ans	Gas price is the amount of ETH a user is prepared to pay for each unit of gas. The user sets the gas price for his transaction depending on how high the priority of a transaction is.

Q25	What are fallback functions in Solidity and how do you write them?

Ans

a. Fallback functions are functions which are executed if a transaction is made to the smart contract without defining any function signature.
A solidity fallback function executes in the following cases:
   i. If other functions do not equal the provided identifier, it works on a call to the contract.
   ii. When a contract gets Ether without any other data, the function gets executed (if it is set as Solidity payable).
   iii. If there is enough gas, this function can execute like any other method.

b. contract Sink {
  function() external payable { }
}

Q26

Which of the lines (among I, II, III and IV) will give an error in Solidity?

```
pragma solidity ^0.4.0;

contract Contract1
{
uint public data;
        function a(uint a) internal returns (uint b)
{
return a + 1;
}
function b(uint ret)
{
ret=a(8);                                   // I
}
function c(uint ret)
{
        ret=this.a(8);                              // II
}
}

contract Contract2
```

	`` { function d(address addr) { Contract1 instance=Contract1(addr); uint ret=instance.a(8); // III } } contract Contract3 is Contract1 { function e() { uint ret=a(8); // IV } } ```
Ans	The lines II and III will give an error. Line II because this keyword means an external call.

Q27	What is the use of the 'payable' keyword in Solidity?
Ans	The payable keyword allows a function to receive ether while being called by any other function.

Q28	What is the return value of the following expression if it is written in Solidity and compiled? ``` if(1) { return true; } else { ```

	`` return false; } ```
Ans	There is an error in the code snippet. '1' cannot be typecast to boolean value in Solidity.

Q29	What are the advantages of ERC 223 over ERC 20?
Ans	The advantages of ERC 223 over ERC 20 are as follows: a. ERC 223 eliminates the problem of lost tokens which happens during the transfer of ERC 20 tokens to a contract (when people mistakenly use the instructions for sending tokens to a wallet). ERC 223 allows users to send their tokens to either the wallet or the contract with the same function transfer, thereby eliminating the potential for confusion and lost tokens. b. ERC 223 allows developers to handle incoming token transactions and reject non-supported tokens (not possible with ERC 20). c. Energy savings: The transfer of ERC 223 tokens to a contract is a one-step process as opposed to a two-step process (in ERC 20), and this means two times less gas and no extra blockchain bloating.

Q30	Which of the following is necessary while using *callcode*, *delegatecall*? a. Ordering of variables b. Target address of the contract c. Signature of the function to be called d. All of the above
Ans	d. All of the above

upGrad

Q31	How do you create an eternal storage contract? How do other smart contracts fetch information stored in the external storage?
Reference	https://fravoll.github.io/solidity-patterns/eternal_storage.html
Ans	contract EternalStorage { address owner = msg.sender; address latestVersion; mapping(bytes32 => uint) uIntStorage; mapping(bytes32 => address) addressStorage; modifier onlyLatestVersion() { require(msg.sender == latestVersion); _; } function upgradeVersion(address _newVersion) public { require(msg.sender == owner); latestVersion = _newVersion; } // *** Getter Methods *** function getUint(bytes32 _key) external view returns(uint) { return uIntStorage[_key]; } function getAddress(bytes32 _key) external view returns(address) { return addressStorage[_key]; } // *** Setter Methods *** function setUint(bytes32 _key, uint _value) onlyLatestVersion external { uIntStorage[_key] = _value;

```
    }

    function setAddress(bytes32 _key, address _value)
  onlyLatestVersion external {
      addressStorage[_key] = _value;
    }

    // *** Delete Methods ***
    function deleteUint(bytes32 _key) onlyLatestVersion
  external {
      delete uIntStorage[_key];
    }

    function deleteAddress(bytes32 _key) onlyLatestVersion
  external {
      delete addressStorage[_key];
    }
  }
```

Q32

Consider the following contract:

```
pragma solidity ^0.4.17;

contract BaseContract {
  event callMeEvent(address _from);
  function callMe() payable public {
    callMeEvent(this);
  }
}

contract ThatCallsBaseContract {
  function callTheOtherContract(address _contractAddress) public {
    require(_contractAddress.call(bytes4(keccak256("callMe()"))));
  }
}
```

	What will be the value of variable 'this' upon the function call of *callTheOtherContract(<Base Contract Address>)?* A. Address of Base Contract B. Address of ThatCallsBaseContract C. Address of msg.sender D. None of the above
Ans	A. Address of Base Contract

Q33	What naming conventions are to be followed for defining test contracts in the Truffle suite?
Ans	The contract name should be followed by the 'test' suffix. For example, for testing Ballot.sol, the testing contract name should be BallotTest.sol