Ethereum Theoretical Exercises

Smart Contract Use Cases

(a) There are many use cases for smart contracts. Write down four of them.

- **Token systems**- Tokens are a sub-currency of Ethereum and represent a certain asset. These systems can for example be used for initial coin offerings (ICOs).
- **Identity and reputation systems**- In order to manage identities without a trusted third party, a smart contract can be used.
- **Decentralized Autonomous Organization**- A DAO is an organization that is controlled entirely by multiple smart contracts. It makes all decisions in a transparent way via its code. Its stakeholders can vote on the decision to make.
- **Election and voting systems**- Smart contracts can record votes in a non-modifiable way and can make sure that wallet owners do not vote twice.
- **Gambling**- Smart contracts can be used to create lotteries, collectibles, pyramid schemes, and other games.
- **Token exchanges**- Some exchange create a new smart contact for every user that register for their service instead of having just one large account that holds all assets. That way, there is no single point-of-failure and this approach facilitates user token management.
- **Name registrars**- Name registrars provide human-readable names for addresses, similar to the Domain Name System (DNS) on the Internet.

(b) Find a contract for each use case you have listed in the previous subtask.
Hint: `https://etherscan.io/labelcloud` might help.

- **Token system**- this is the ERC20 contract of Maker: 0x9f8F72aA9304c8B593d555F12eF6589cC3A579A2
- **DAO**-this is the contract of the original DOA: 0xBB9bc244D798123fDe783fCc1C72d3Bb8C189413
- **Gambling**-this it the CryptoKitties smart contract: 0x06012c8cf97BEaD5deAe237070F9587f8E7A266d
- **Cryptocurrency exchange**- this is a smart contract of the cryptocurrency exchange Uniswap: 0x5C69bEe701ef814a2B6a3EDD4B1652CB9cc5aA6f

EVM and Gas

(a) What is gas in the context of Ethereum?

> Every instruction of the EVM has a certain cost assigned, called gas. In general, more complex instructions cost more gas because they consume more computing power of the nodes in the network. A sender of a transaction sets a price in Ether that they are willing to pay per gas unit and a maximum amount of gas they want to spend. The transaction fee that the miner of the transaction gets to keep is the product of used gas units and gas price.

(b) What happens with spare gas if too much gas was provided for a transaction?

> If a transaction uses less gas than the sender provided, the excess amount gets refunded to the sender.

(c) Name on relatively expensive and one relatively cheap EVM instruction and briefly explain why. You can use `https://ethereum.org/en/developers/docs/evm/opcodes/`.

> - Expensive
>   - **CALL**: Invoke functions on other contracts or send Ether to them (dynamic cost - `https://github.com/wolflo/evm-opcodes/blob/main/gas.md#aa-call-operations`)
>   - **CREATE**: Create other contracts (32000 gas)
> - Cheap
>   - **STOP**: Halts execution (0 gas)

(d) Calculate an estimation of how much it approximately costs (in USD) to store 1 KiB of data on the Ethereum blockchain at the moment. For simplicity, consider only the cost of SSTORE operations and disregard everything else. Use current average gas price from Etherscan for your calculation.

> A word in Ethereum is 32 bytes long. For 1024 bytes there are 1024/32=32 SSTORE operations necessary as every operation stores one word on the blockchain. We multiply the gas cost by the gas price in ETH and by the price of ETH in USD: 32 * 20000 * 0.00000002 * 1149.81= 14.71 USD

(e) Name one instruction that can refund gas.

- **SELFDESTRUCT**: Destroy the contract (max 24000 gas refunded)
- **SSTORE[x] = 0**: Clear storage space (max 15000 gas refunded)

Storing Data in Smart Contracts

(a) At a very low level, storing data to storage is performed by the EVM instruction SSTORE. According to the yellow paper, the gas cost of an SSTORE operation is 20,000 when storage value is set to non-zero from zero and 5,000 in any other case. Why is it more expensive to set a value from zero to non-zero?

> All storage items of a contract are initially empty. Setting a storage value from 0 to non-zero means that this value is newly initialized and all full nodes in the network have to keep track of the new data item. Every client has to create a new database entry for this item, which increases the size of the world state. As every node in the network has to allocate physical hard disk space, this operation is quite expensive. Once that space is allocated, changing the value is not increasing the size of the state, which is why this operation is cheaper.

Transactions in Ethereum

(a) In Ethereum there are two types of transactions. Whenever a wallet calls a method on a contract, a transaction is sent. Whenever a contract calls a method on another contract, a virtual message is sent (Sometimes referred as internal transactions). Messages exist "virtually" and are not written to the blockchain. Why are messages not published to blockchain? How could the blockchain be in the same state among all nodes without writing messages to the blockchain?

> The Ethereum Yellow Paper describes transactions as "Transaction: A piece of data, signed by an External Actor." Think of the Ethereum VM as a closed system and wallets interacting with the system and therefore changing the state. Ethereum is based on strict determinism. The state of the VM must be same across all nodes at a certain point. To reach the most recent state all nodes run the Ethereum VM and process mined blocks, which include transactions. Take a new node that just enters the system. First thing the node has to do is to sync with the current state of the blockchain. To achive this the node has to start from the genesis block and go through all blocks until the most recent one. During these steps the node will execute all transactions and also the messages resulting from transactions. In other words the node simulates all interactions of other wallets and all messages these interactions trigger. So each and every message will eventually be executed by the syncing node. Therefore there is no need to write the messages to the blockchain as the information is inherently contained in the transactions.

(b) Name four reasons why a transaction that is sent to the Ethereum network might not get mined.

> i. The transaction has a wrong signature.
> ii. Sending account does not have enough funds.
> iii. Sender set a too low gas price to be added by a miner.
> iv. Transaction consumes more gas than the block gas limit.

(c) Name two different ways how a mined transaction can fail.

> i. The transaction runs out of gas.
> ii. The transaction gets reverted (e.g. by executing an illegal instruction, the REVERT instruction, or a JUMP to an illegal destination).

(d) Why does a mined transaction that fails still cost gas?

> In order to determine that a transaction fails, it has to be executed. At the point of failure, the transaction has already consumed computing power. Therefore, the sender should pay for it. This measure prevents spam. If a failed transaction would not cost any gas, a spammer could send many transactions with complex instructions that fail in the end. The spammer would not be charged but the transactions consume a lot of computing power from the members of the network.

True/False

1. Are these statements true or false? Give a short explanation.

   (a) Every pure function is a view function.

   > True. Pure functions are a subset of view functions which don't modify the state but also don't read from the state.

   (b) In Ethereum, a smart contract that has no functions that is declared as **payable** cannot receive any Ether.

   > False. It can receive Ether as result of a coinbase transaction or as a destination of a selfdestruct.

   (c) If **msg.sender == tx.origin**, then this code was called directly by an externally owned account and not by another smart contract.

   > True. **tx.origin** is the externally owned account that issued the transaction and **msg.sender** is the direct caller of the function. If they are equal, the code was called directly by the issuer of the transaction and therefore not by a smart contract.