Blockchains and Distributed Ledgers - Coursework 3

s1732368

January 18, 2021

1 Part 1

Key: ;)
I went to the website https://ropsten.etherscan.io/ and searched for the contract's address. I then looked at the successful registration transactions and was able to find the key by decoding the input data. I used this key for my registration transaction.

2 Part 2a

safemath : 0x9C91b7FB11133483D81Aab62B7713E29aF0212a2
contract : 0x9329Df9dE3fDc99B9ecF570A0e7cA928E3DC502d

2.1 High-level design decisions

I use a mapping, balance, to associate the address of transactions with their token balance.
I also have a creator variable to store the contract owner.
I use a uint256 variable, soldTokens, to store the number of tokens that have been sold and to ensure the contract has enough funds to buy them back.

To buy a token, first it is checked if the amount of wei that is sent is at least the token price * the number of tokens the user would like to purchase. If this is true then the balance of the user is increased to reflect their newly purchased tokens and any additional wei is refunded to the user. A Purchase event is emitted and true is returned.

To sell a token, it is first checked that the user is not attempting to sell more tokens than they own. The user's balance is then decreased and they are sent wei equal to the amount of tokens they are selling * the tokenPrice. A Sell event is emitted and true is returned.

To transfer a token is similar to selling, where a check is performed to ensure the user is transferring tokens which they own. The balance of the user who is performing the transfer is decreased by the transfer amount and the balance of the user who is receiving the tokens is increased. A Transfer event is emitted and true is returned.

To increase the price of a token, a check is performed to ensure the user is the owner of the contract. Then, another check is carried out to see if enough wei has been sent in the transaction to ensure the contract can still function correctly with a price increase. After both checks pass, the price is altered, a Price event is emitted and true is returned.

To view their balance, a user would call the getBalance function.

To link the safe math library to my contract, I called it in my cw3.sol file and associated it with uint256s. Then, I deployed the SafeMath library on the network and saved the address. I altered the metadata file of my contract after I compiled it and added the SafeMath library address so that it would know where to look after I deployed it.

2.2 Gas evaluation

Deployment costs: 1115225 gas

	Transaction	Execution	Total
buyToken	71854	50390	122244
transfer	41939	34067	76006
sellToken	38516	32052	70568
changePrice	31667	10203	41870
getBalance	0	0	0
Total	**183976**	**126712**	**310688**

In my contract I used uint256 data types for int storage as they are more gas efficient than smaller uint data types. All functions that are only called from outside the contract are marked as 'external', as this saves on gas when called[2].

When performing checks within functions, I always used require() instead of assert() as the former is more gas efficient[1].

Future versions could add the safe math library code directly into the contract as this can save on gas[1].

When compiling my contract, I enabled optimisations to hopefully save gas as well.

2.3 Potential hazards and vulnerabilities

The contract could be vulnerable to overflow or underflow attacks, but the Safe Math library should help prevent these vulnerabilities from occurring.

When buying tokens, the user could attempt to buy more than they could afford with the wei they have sent. To prevent this, I added a require statement to check for this.

When transferring tokens, the user could attempt to transfer more than they own. To prevent this, I added a require statement to check for this.

When selling tokens, the user could attempt to sell more than they own, allowing them to get more wei than they should be able to. To prevent this, I added a require statement to check for this.

To send wei to the user, I use the transfer function to help prevent against reentrancy attacks.

When changing the token price, a user who is not the contract owner could attempt to do this, allowing them to gain additional wei. To prevent this I have added a require statement to ensure the user carrying out the transaction is the contract creator.

When increasing the token price, it is possible that the contract will not have enough wei and will not be able to buy back all of the tokens that have been sold to users. To prevent this from happening, a check has been added to ensure the owner sends enough wei with the price increase so that the contract can afford all distributed tokens.

In case of some unforeseen bug in my contract, I have added a check when selling tokens to ensure the contract has enough wei to buy all that the user is selling. This ensures that the user is not robbed of wei when they sell their tokens. I have also added a separate payable function to top up the contract in case of any bugs as well.

2.4 Transaction history of deployment


```
{
  "accounts": {
    "account{0}": "0x79Ea5f0Ec96A8d94087D9Bf58ce4A848F529EB0E"
  },
  "linkReferences": {
    "SafeMath": "<address>"
  },
  "transactions": [
    {
      "timestamp": 1610898926847,
      "record": {
        "value": "0",
        "parameters": [],
        "abi": "0xa9c9e2dd513faefff720240af77f7e67c03d87ed906192b844da3ce7b2484fc0",
        "contractName": "cw3",
        "bytecode": "60806040526001600055348015610015576000 80fd5b50336001600061000a81548173ffffffffffffffffffffffffffffffffffffffff02191690837 3ffffffffffffffffffffffffffffffffffffffff1602179055506112ba80610066000396000f3fe608060405260043610610059576003560e01c806312065fe0146100655780632397e4d7146100905780632d296bf1146100e15780637ff9b596146101255578063a2b40d1914610150578063a9059cbb14610194576100 60565b3661006057005b600080fd5b348015610071576000 80fd5b50610078610215565b604051808281526020019150506040
```

```
05180910390f35b34801561009c57600080fd5b506100c96004
80360360208110156100b357600080fd5b81019080803590602
00190929190505061024c565b604051808215158152602001
91505060405180910390f35b61010d60048036036020811015
100f757600080fd5b810190808035906020019092919050505
6106c5565b604051808215158152602001915050604051809
390f35b34801561013157600080fd5b5061013a610b42565b60
40518082815260200191505060405180910390f35b61017c60
48036036020811015610166576000 80fd5b810190808035906
2001909291905050610b48565b604051808215158152602
19150506040518091039 0f35b3480156101a057600080fd5b50
6101ed60048036036040811015610 1b757600080fd5b8101908
0803573ffffffffffffffffffffffffffffffffffffffff1690
602001909291908035906020019092919050505061 0d2b565b6
040518082151581526020019150506040518091039 0f35b6000
600360003373
ffffffffffffffffffffffffffffffffffffffff1673
ffffffffffffffffffffffffffffffffffffffff16815260200
190815260200160002054905090565b600080600360003373
ffffffffffffffffffffffffffffffffffffffff1673
ffffffffffffffffffffffffffffffffffffffff16815260200
190815260200160002054905080831115 6102ea576040517f08
c379a0000000000000000000000000000000000000000000000
0000000008152600401808060200182810382526 03d815260
20018061119c603d9139604001915050604051809 10390fd5b4
78373__$3060793c0de70aceca490b044bfa4393b2$__63c8a4
ac9c909160005460405183 63ffffffff1660e01b81526004018
083815260200182815260200192505050602060405180830381
86803b15801561034657600080fd5b505af415801561035a573
d6000803e3d6000fd5b505050506040513d6020811015610370
57600080fd5b810190808051906020019092919050505011156
103d8576040517f08c379a0000000000000000000000000000
0000000000000000000000000000081526004018080602001828
1038252604281526020018061112860429139606001915050 60
405180910390fd5b600360003373
ffffffffffffffffffffffffffffffffffffffff1673
ffffffffffffffffffffffffffffffffffffffff16815260200
190815260200160002054 73__$3060793c0de70aceca490b044
bfa4393b2$__63b67d77c59091856040518363ffffffff1660e
01b8152600401808381526020018281526020019250505060 20
60405180830381 86803b15801561047057600080fd5b505af41
58015610484573d6000803e3d6000fd5b505050506040513d60
2081101561049a57600080fd5b810190808051906020019092 9
19050505060036000 3373
ffffffffffffffffffffffffffffffffffffffff1673
ffffffffffffffffffffffffffffffffffffffff16815260200
```

```
1908152602001600020819055506002547 3__$3060793c0de70
aceca490b044bfa4393b2$__63b67d77c59091856040518363
ffffffff1660e01b81526004018083815260200182815260200
192505050602060405180830381868 03b158015610549576000
80fd5b505af415801561055d573d6000803e3d6000fd5b50505
0506040513d602081101561057357600080fd5b810190808051
906020019092919050505060028190555033 73
ffffffffffffffffffffffffffffffffffffffff166108fc847
3__$3060793c0de70aceca490b044bfa4393b2$__63c8a4ac9c
909160005460405183 6ffffffff1660e01b815260040180838
152602001828152602001925050506020604051808303818680
3b1580156105ff57600080fd5b505af4158015610613573d600
0803e3d6000fd5b505050506040513d6020811015610629576 0
0080fd5b81019080805190602001909291905050509081150 29
060405160006040518083038185888 8f1935050505015801561
0665573d6000803e3d6000fd5b507f5e5e995ce3133561
afceaa51a9a154d5db228cd7525d34df5185582c18d3df09338
4604051808373
ffffffffffffffffffffffffffffffffffffffff16815260200
182815260200192505050604051809103 90a160019150509190
50565b60008060005473__$3060793c0de70aceca490b044bfa
4393b2$__63c8a4ac9c9091856040518363ffffffff1660e01b
815260040180838152602001828152602001925050506020604
0518083038186803b158015610723576 00080fd5b505af41580
15610737573d6000803e3d6000fd5b505050506040513d60208
1101561074d57600080fd5b810190808051906020019092919 0
505050908034101561 07b9576040517f08c379a0000000000
0000000000000000000000000000000000000000000008152
600401808060200182810382526 02f8152602001806110f9602
f9139604001915050604051809103 90fd5b600360003373
ffffffffffffffffffffffffffffffffffffffff1673
ffffffffffffffffffffffffffffffffffffffff16815260200
190815260200160002054 73__$3060793c0de70aceca490b044
bfa4393b2$__63771602f79091856040518363ffffffff1660e
01b815260040180838152602001828152602001925050506020
604051808303818 6803b158015610851576 00080fd5b505af41
58015610865573d6000803e3d6000fd5b505050506040513d60
2081101561087b57600080fd5b810190808051906020019092 9
190505050600360003373
ffffffffffffffffffffffffffffffffffffffff1673
ffffffffffffffffffffffffffffffffffffffff16815260200
190815260200160002081905550600254 73__$3060793c0de70
aceca490b044bfa4393b2$__63771602f79091856040518363
ffffffff1660e01b815260040180838152602001828152602001
925050506020604051808303818 6803b15801561092a576000
80fd5b505af415801561093e573d6000803e3d6000fd5b50505
```

```
0506040513d60208110156109545760008 0fd5b810190808051
90602001909291905050506002819055506000547 3__$306079
3c0de70aceca490b044bfa4393b2$__63c8a4ac9c9091856040
518363ffffffff1660e01b8152600401808381526020018281 5
26020019250505060206040518083038186803b1580156109c6
57600080fd5b505af41580156109da573d6000803e3d6000fd5
b505050506040513d60208110156109f057600080fd5b810190
8080519060200190929190505050341115610ae3573 73
ffffffffffffffffffffffffffffffffffffffff166108fc347
3__$3060793c0de70aceca490b044bfa4393b2$__63b67d77c5
9091856040518363ffffffff1660e01b8152600401808381526
020018281526020019250505060206040518083038186803b15
8015610a7b57600080fd5b505af4158015610a8f573d6000803
e3d6000fd5b505050506040513d6020811015610aa557600080
fd5b810190808051906020019092919050505090811502906 04
0516000604051808303818588 8f193505050158015610ae1
573d6000803e3d6000fd5b505b7f2499a5330ab0979cc612135
e7883ebc3cd5c9f7a8508f042540c34723348f6323384604051
808373ffffffffffffffffffffffffffffffffffffffff16815
260200182815260200192505050604051809103 90a160019150
50919050565b60005481565b60006001600090549061010 00a9
00473ffffffffffffffffffffffffffffffffffffffff1673
ffffffffffffffffffffffffffffffffffffffff163373
ffffffffffffffffffffffffffffffffffffffff1614610bf05
76040517f08c379a0000000000000000000000000000000000
0000000000000000000008152600401808060200182810382 5
260328152602001806111 6a6032913960400191505060405180
910390fd5b60008273__$3060793c0de70aceca490b044bfa43
93b2$__63c8a4ac9c909160025460405183 63ffffffff1660e0
1b8152600401808381526020018281526020019250505060206
040518083038186803b158015610c4d57600080fd5b505af415
8015610c61573d6000803e3d6000fd5b505050506040513d602
0811015610c7757600080fd5b810190808051906020019092 91
905050509050478111156 10ce3576040517f08c379a00000000
0000000000000000000000000000000000000000000000081
526004018080602001828103825260458152602001806111d96
045913960600191505060405180910390fd5b82600081905550
7f63f32f63810afda7c9be9643f9fa73ee3f39a9fd8bb35775a
2b0d73e48ed9bed836040518082815260200191505060405180
910390a1600191505091905056 5b600080600360003373
ffffffffffffffffffffffffffffffffffffffff1673
ffffffffffffffffffffffffffffffffffffffff16815260200
190815260200160002054905080831115610dc9576040517f08
c379a0000000000000000000000000000000000000000000000
0000000008152600401808060200182810382526041815260
20018061121e6041913960600191505060405180910390fd5b3
```

```
373ffffffffffffffffffffffffffffffffffffffff168473
ffffffffffffffffffffffffffffffffffffffffff161415610e4
e576040517f08c379a0000000000000000000000000000000000
0000000000000000000000008152600401808060200182810382
5260268152602001806112 5f6026913960400191505060405180910390fd5b600360003373
ffffffffffffffffffffffffffffffffffffffff1673
ffffffffffffffffffffffffffffffffffffffffff16815260200
190815260200160002054 73__$3060793c0de70aceca490b044bfa4393b2$__63b67d77c59091856040518363ffffffff1660e
01b8152600401808381526020018281526020019250505060200
6040518083038186803b158015610ee657600080fd5b505af41
58015610efa573d6000803e3d6000fd5b505050506040513d60
20811015610f1057600080fd5b810190808051906020019092919050505060036000 3373
ffffffffffffffffffffffffffffffffffffffff1673
ffffffffffffffffffffffffffffffffffffffffff16815260200
190815260200160002081905550600360008573
ffffffffffffffffffffffffffffffffffffffff1673
ffffffffffffffffffffffffffffffffffffffffff16815260200
190815260200160002054 73__$3060793c0de70aceca490b044bfa4393b2$__63771602f79091856040518363ffffffff1660e
01b8152600401808381526020018281526020019250505060200
6040518083038186803b158015610ffc57600080fd5b505af41
58015611010573d6000803e3d6000fd5b505050506040513d60
208110156110265760008 0fd5b810190808051906020019092919050505060036000 8673
ffffffffffffffffffffffffffffffffffffffff1673
ffffffffffffffffffffffffffffffffffffffffff16815260200
190815260200160002081905550 7fddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef3385856040
51808473ffffffffffffffffffffffffffffffffffffffff16
81526020018373
ffffffffffffffffffffffffffffffffffffffffff16815260200
182815260200193505050506040518091039 0a1600191505092
91505056fe506c656173652073656e6420656e6f75676820776
56920746f2070757263686173652074686573652074 6f6b656e
7354686520636f6e7472616374206 46f6573206e6f742068617
66520656e6f756768207765692074 6f20627579207468657365
20746f6b656e732066726f6d20796f754f6e6c7920746865206
36f6e7472616374206f776e65722063616e206d6f6469667920
74686520746f6b656e207072696365596f75206172652061747
4656d7074696e6720746f2073656c6c206d6f726520746f6b65
6e73207468616e20796f752063757272656e746c79206f776e5
06c656173652073656e64206164646974696f6e616c20776569
20746f20656e737572652074686520636f6e747261637420667
```

```
            56e6374696f6e7320636f72726563746c79596f752061726520
            617474656d7074696e6720746f207472616e73666572206d6f7
            26520746f6b656e73207468656e20796f752063757272656e74
            6c79206f776e596f752063616e6e6f74207472616e736665722
            0746f6b656e7320746f20796f757273656c66a2646970667358
            2212208a3ae5166452e01a6df2a0307ef19858ad9dfc6b3943f
            74d2a796fd0bb82c5da64736f6c634300060c0033",
        "linkReferences": {
          "SafeMath.sol": {
            "SafeMath": [
              {
                "length": 20,
                "start": 852
              },
              {
                "length": 20,
                "start": 1152
              },
              {
                "length": 20,
                "start": 1369
              },
              {
                "length": 20,
                "start": 1549
              },
              {
                "length": 20,
                "start": 1843
              },
              {
                "length": 20,
                "start": 2145
              },
              {
                "length": 20,
                "start": 2362
              },
              {
                "length": 20,
                "start": 2518
              },
              {
                "length": 20,
                "start": 2699
              },
```

```
            {
              "length": 20,
              "start": 3163
            },
            {
              "length": 20,
              "start": 3830
            },
            {
              "length": 20,
              "start": 4108
            }
          ]
        }
      },
      "name": "",
      "inputs": "()",
      "type": "constructor",
      "from": "account{0}"
    }
  }
],
"abis": {
  "0xa9c9e2dd513faefff720240af77f7e67c03d87ed906192b844da3ce7b2484fc0": [
    {
      "anonymous": false,
      "inputs": [
        {
          "indexed": false,
          "internalType": "uint256",
          "name": "price",
          "type": "uint256"
        }
      ],
      "name": "Price",
      "type": "event"
    },
    {
      "anonymous": false,
      "inputs": [
        {
          "indexed": false,
          "internalType": "address",
          "name": "buyer",
          "type": "address"
```

```
            },
            {
              "indexed": false,
              "internalType": "uint256",
              "name": "amount",
              "type": "uint256"
            }
          ],
          "name": "Purchase",
          "type": "event"
        },
        {
          "anonymous": false,
          "inputs": [
            {
              "indexed": false,
              "internalType": "address",
              "name": "seller",
              "type": "address"
            },
            {
              "indexed": false,
              "internalType": "uint256",
              "name": "amount",
              "type": "uint256"
            }
          ],
          "name": "Sell",
          "type": "event"
        },
        {
          "anonymous": false,
          "inputs": [
            {
              "indexed": false,
              "internalType": "address",
              "name": "sender",
              "type": "address"
            },
            {
              "indexed": false,
              "internalType": "address",
              "name": "receiver",
              "type": "address"
            },
            {
```

```
            "indexed": false,
            "internalType": "uint256",
            "name": "amount",
            "type": "uint256"
          }
        ],
        "name": "Transfer",
        "type": "event"
      },
      {
        "inputs": [
          {
            "internalType": "uint256",
            "name": "amount",
            "type": "uint256"
          }
        ],
        "name": "buyToken",
        "outputs": [
          {
            "internalType": "bool",
            "name": "",
            "type": "bool"
          }
        ],
        "stateMutability": "payable",
        "type": "function"
      },
      {
        "inputs": [
          {
            "internalType": "uint256",
            "name": "price",
            "type": "uint256"
          }
        ],
        "name": "changePrice",
        "outputs": [
          {
            "internalType": "bool",
            "name": "",
            "type": "bool"
          }
        ],
        "stateMutability": "payable",
        "type": "function"
```

```
    },
    {
      "inputs": [
        {
          "internalType": "uint256",
          "name": "amount",
          "type": "uint256"
        }
      ],
      "name": "sellToken",
      "outputs": [
        {
          "internalType": "bool",
          "name": "",
          "type": "bool"
        }
      ],
      "stateMutability": "nonpayable",
      "type": "function"
    },
    {
      "inputs": [
        {
          "internalType": "address",
          "name": "recipient",
          "type": "address"
        },
        {
          "internalType": "uint256",
          "name": "amount",
          "type": "uint256"
        }
      ],
      "name": "transfer",
      "outputs": [
        {
          "internalType": "bool",
          "name": "",
          "type": "bool"
        }
      ],
      "stateMutability": "nonpayable",
      "type": "function"
    },
    {
      "stateMutability": "payable",
```

```
        "type": "receive"
      },
      {
        "inputs": [],
        "stateMutability": "nonpayable",
        "type": "constructor"
      },
      {
        "inputs": [],
        "name": "getBalance",
        "outputs": [
          {
            "internalType": "uint256",
            "name": "",
            "type": "uint256"
          }
        ],
        "stateMutability": "view",
        "type": "function"
      },
      {
        "inputs": [],
        "name": "tokenPrice",
        "outputs": [
          {
            "internalType": "uint256",
            "name": "",
            "type": "uint256"
          }
        ],
        "stateMutability": "view",
        "type": "function"
      }
    ]
  }
}
```


2.5 Transaction history of fellow student's contract


```
{
  "accounts": {
    "account{0}": "0x79Ea5f0Ec96A8d94087D9Bf58ce4A848F529EB0E"
  },
  "linkReferences": {},
  "transactions": [
```

```
    {
      "timestamp": 1610898521393,
      "record": {
        "value": "20",
        "parameters": [
          "10"
        ],
        "to": "created{undefined}",
        "name": "buyToken",
        "inputs": "(uint256)",
        "type": "function",
        "from": "account{0}"
      }
    },
    {
      "timestamp": 1610898545562,
      "record": {
        "value": "0",
        "parameters": [
          "3"
        ],
        "to": "created{undefined}",
        "name": "changePrice",
        "inputs": "(uint256)",
        "type": "function",
        "from": "account{0}"
      }
    },
    {
      "timestamp": 1610898552418,
      "record": {
        "value": "0",
        "parameters": [
          "5"
        ],
        "to": "created{undefined}",
        "name": "sellToken",
        "inputs": "(uint256)",
        "type": "function",
        "from": "account{0}"
      }
    },
    {
      "timestamp": 1610898579504,
      "record": {
        "value": "0",
```

```
        "parameters": [
          "0x7ce0247DEeEA78546FfE02864Cae0e519be31e2B",
          "4"
        ],
        "to": "created{undefined}",
        "name": "transfer",
        "inputs": "(address,uint256)",
        "type": "function",
        "from": "account{0}"
      }
    }
  ],
  "abis": {}
}
```

2.6 The code of my contract

```
pragma solidity >=0.4.22 <=0.8.0;
// Import the SafeMath library
import "SafeMath.sol";

contract cw3{

        // Use SafeMath functions for uint256s
        using SafeMath for uint256;

        // Store the tokenPrice
        uint256 public tokenPrice = 1;

        // Stores the owner of the contract
        address private creator;

        // Stores the number of tokens sold
        uint256 private soldTokens;

        // Maps user addresses to their balance
        mapping(address => uint256) private balance;

        // Events for the different transactions
        event Purchase(address buyer,uint256 amount);
        event Transfer(address sender,address receiver,uint256 amount);
        event Sell(address seller, uint256 amount);
        event Price(uint256 price);

        // The constructor stores the contract's owner
        constructor() public {
            creator = msg.sender;
        }

        // Function to buy a token
        function buyToken(uint256 amount) external payable returns(bool){
            uint256 cost = tokenPrice.mul(amount);
```

```
36              // Check is made to ensure the wei sent is at least that required
                    to buy a token
37              require(msg.value >= cost, "Please send enough wei to purchase
                    these tokens");
38              // Increases the user's balance to reflect their newly purchased
                    tokens
39              balance[msg.sender] = balance[msg.sender].add(amount);
40              // Increment the number of tokens sold
41              soldTokens = soldTokens.add(amount);
42              // Refund extra wei
43              if(msg.value > tokenPrice.mul(amount)){
44                  msg.sender.transfer(msg.value.sub(cost));
45              }
46
47              // Emit a Purchase event and return true
48              emit Purchase(msg.sender, amount);
49              return true;
50          }
51
52          // Function to transfer tokens to another user
53          function transfer(address recipient, uint256 amount) external returns
                (bool){
54              uint256 senderBalance = balance[msg.sender];
55              // Check if the user has enough tokens in their account to send
                    the number specified
56              require(amount <= senderBalance, "You are attempting to transfer
                    more tokens then you currently own");
57              // Check if the user is transferring tokens to themselves
58              require(recipient != msg.sender, "You cannot transfer tokens to
                    yourself");
59              // Decrease the sender's balance
60              balance[msg.sender] = balance[msg.sender].sub(amount);
61              // Increase the receiver's balance
62              balance[recipient] = balance[recipient].add(amount);
63              // Emit a transfer event and return true
64              emit Transfer(msg.sender, recipient, amount);
65              return true;
66          }
67
68          // Function to sell tokens
69          function sellToken(uint256 amount) external returns(bool){
70              uint256 senderBalance = balance[msg.sender];
71              // Check if the user has enough tokens in their account to seel
                    the number specified
72              require(amount <= senderBalance, "You are attempting to sell more
                     tokens than you currently own");
73              // Check if the contract has enough wei to send to the user
74              require(amount.mul(tokenPrice) <= address(this).balance, "The
                    contract does not have enough wei to buy these tokens from
                    you");
75              // Decrease the user's balance
76              balance[msg.sender] = balance[msg.sender].sub(amount);
77              // Decrease the number of sold tokens
78              soldTokens = soldTokens.sub(amount);
79              // Send the user wei equal in value to the tokens they have sold
80              msg.sender.transfer(amount.mul(tokenPrice));
81              // Emit a Sell event and return true
82              emit Sell(msg.sender, amount);
```

```
            return true;
        }

        // Function for owner to change the token price
        function changePrice(uint256 price) external payable returns(bool){
            // Check if the person calling this function is the owner of the
                contract
            require(msg.sender == creator, "Only the contract owner can
                modify the token price");
            // Ensure the contract has enough wei in the case of a price
                increase
            uint256 weiNeeded = price.mul(soldTokens);
            require(weiNeeded <= address(this).balance, "Please send
                additional wei to ensure the contract functions correctly");
            // Set the token price to the value the user has specified
            tokenPrice = price;
            // Emit a Price event and return true
            emit Price(price);
            return true;
        }

        // Function to check the user's balance
        function getBalance() external view returns(uint256){
            // Return the balance associated with the user's address
            return balance[msg.sender];
        }

        // Function for contract to receive payments
        receive() external payable{

        }

}
```

3 Part 2b

This could not be entirely completed on chain. First of all, the identification document would need to be sent to the issuer, along with your Ethereum address. The issuer would then reply with a key and upload a hash of the key to the smart contract, associating it with your Ethereum address via a mapping.

Then, when purchasing a token, the user would provide the key along with the number of tokens they wish to buy. The buy function would then hash the key the user sends and and check it against the mapping, confirming the user's identity.

4 References

[1] https://medium.com/layerx/how-to-reduce-gas-cost-in-solidity-f2e5321e0395

[2] https://medium.com/better-programming/how-to-write-smart-contracts-that-optimize-gas-spent-on-ethereum-30b5e9c5db85

[3] https://eattheblocks.com/how-to-optimize-gas-cost-in-a-solidity-smart-contract-6-tips/