CSIT6000Q (L1)
Fall 2023
16/12/2023
Blockchain and Smart Contracts
Final Exam Question Paper
Time Limit: 90 Minutes

Name: _____
Student ID: _____
Email Address: _____

Total number of pages: 24 (including this cover page)
Total number of questions: 7
Total score: 147

Grade Table (for teacher use only)

Question	Points	Score
1	30	
2	34	
3	20	
4	10	
5	21	
6	18	
7	14	
Total:	147	

Question 1 [30 points]

In each question, please select only ONE correct answer, unless mentioned otherwise explicitly.

(a) (1 point) Which one of the following is not an application of hash functions?

A. One-way password file.

B. Virus detection.

C. Intrusion detection.

D. Key wrapping.

(b) (1 point) When a hash function is used to provide message authentication, the hash function value is referred to as

A. Message Field

B. Message Score

C. Message Digest

D. Message Leap

(c) (1 point) An orphan block is only created when 51% attack is successful

A. True.

B. False.

(d) (3 points) H is a hash function and for input X, it maps an output $H(X)$. Which of the following properties does H satisfy? (can have multiple correct options)

A. Efficiently computed.

B. $\forall X_1, X_2, Length(H(X_1)) = Length(H(X_2))$, when $X_1 == X_2$ and X_1 and X_2 represents inputs messages and $Length(X)$ return the length of input X.

C. Deterministic: $\forall X_1, X_2, H(X_1) = H(X_2)$, where $X_1 == X_2$ and X_1 and X_2 represents inputs messages.

D. It is infeasible to determine X from $H(X)$

E. Change X slightly and $H(X)$ changes significantly

F. Even if know $Hash(X)$ and part of X it is still very hard to find rest of X.

(e) (1 point) Which of the following is an example of a hash function?

A. SHA-1.

B. RSA.

C. AES.

D. Diffie-Hellman

(f) (1 point) Which algorithm provides the private key and its corresponding public key?

A. Key generation algorithm.

B. Signature verifying algorithm.

C. Signing algorithm.

D. None of the above

(g) (1 point) A digital signature needs a/an _____ system

A. Symmetric key.
B. Asymmetric key.
C. Either (A) or (B).
D. Neither (A) nor (B)

(h) (1 point) Which of the following can the digital signature not provide for the message?

A. Integrity
B. Confidentiality.
C. Non-repudiation.
D. Authentication

(i) (1 point) What feature about enterprise blockchain is accurate?

A. Is relatively inexpensive.
B. Has trust problems
C. Requires no change management
D. Is energy-efficient
E. All of the above.

(j) (1 point) What is a blockchain?

1 - A Currency
2 - A centralized ledger
3 - A type of cryptocurrency
4 - A distributed ledger on a peer to peer network

A. 1
B. 2
C. 3
D. 4
E. All of them.
F. Both 1 and 3
G. Both 1, 2 and 3
H. Both 1, 2 and 4

(k) (1 point) What is a node?

A. A Blockchain
B. An exchange
C. A type of cryptocurrency
D. A computer on a Blockchain network

(l) (1 point) What is a genesis block?

A. The 2nd transaction of a Blockchain.

B. The first block after each block halving.
C. The first block of a Blockchain.
D. A famous block that hardcoded a hash of the Book of Genesis onto the blockchain.

(m) (1 point) What does the block in the blockchain consist of?
A. Transaction data
B. A Hash point
C. A Timestamp
D. All of these

(n) (3 points) Hash Functions used in Merkle Trees?
A. RSA
B. SHA-128
C. SHA-256
D. RIPEMD160
E. All of these

(o) (3 points) Does the following program have a compile error?

```
pragma solidity >=0.8.2 <0.9.0;
contract studentInfo{
    uint16 constant public studentID;
    uint32 public class;

    function setStudentID(string _studentID) public{
        studentID = _studentID;
    }
    function getStudentID() public pure returns(uint16){
        return studentID;
    }
    function setClass(uint32 _class) public{
        class = _class;
    }
    function getClass() public pure returns(uint32){
        return class;
    }
}
```

A. No, it does not have an error.
B. Yes, it has a compile error.

(p) (3 points) Does the following program have a compile error?

```
pragma solidity >=0.8.2 <0.9.0;

interface MoblieOS { }
interface PowerFullChipset { }
```

```
contract Phone is MoblieOS { }
contract IPhone is Phone, PowerFullChipset{ }
contract Test1 {

       uint16 public price;

       function TestPhone() public{
            Phone ph = new Phone();
            Phone ph2 = new IPhone();
            IPhone iph = new IPhone();
       }
 }
```

A. No, it does not have an error.

B. Yes, it has a compile error.

(q) (3 points) Does the following program have a compile error?

```
pragma solidity >=0.8.2 <0.9.0;
contract Test1 {
     uint32 public testVar1;

    function setClass(uint32 _test) public{
        testVar1 = _test;
    }
    function getClass() public pure returns(uint32){
        return testVar1;
    }
       function funTest1() public{
            if (((x+y)+3) = 2*(y+5)+7) {
                testVar1=23;
            }
            else{
                testVar1=2*23;
            }
       }
 }
```

A. No, it does not have an error.

B. Yes, it has a compile error.

(r) (3 points) Does the following program have a compile error?

```
pragma solidity >=0.8.2 <0.9.0;
contract ArrayTest {

       function ArrayTestFun() public{
           uint[] f1;
           uint[] f2;
```

```
            f1 = new uint[](10);
            f2 = f1;
        }
}
```

A. No, it does not have an error.

B. Yes, it has a compile error.

Question 2 [34 points]

(a) (5 points) Which of the following statements are true?

```
pragma solidity >=0.8.2 <0.9.0;

contract contract_A {
    uint private AX;
    uint public BX;
    string internal Cstr;
    constructor() {
        BX=24;
    }
    function increment(uint Dr) private pure returns(uint){
        return Dr+1;
    }
    function updateAX(uint Dr) public{
        AX=Dr;
    }
    function getAX() public view returns(uint){
        return AX;
    }
    function setCstr(string memory _Cstr) public;
    function getCstr() public view returns(string memory);
}

contract contract_B is contract_A{
    function setCstr(string memory _Cstr) public{
        Cstr = _Cstr;
    }
    function getCstr() public view returns(string memory){
        return Cstr;
    }
}
contract contract_D
{
    function readData() public payable returns(string memory, uint)
    {
        contract_A cA = new contract_A();
        cA.setCstr('Question 2a');
```

```
        cA.updateAX(25);
        return (cA.getAX(), cA.getCstr());
    }
}
```

A. private increment function defined in `contract_A` is accessible in derived contract `contract_B` but not accessible outside the derived contracts.

B. private increment function defined in `contract_A` is accessible only inside `contract_A` and not accessible to derived contracts as well.

C. internal variable `Cstr` defined in `contract_A` is accessible in derived contract `contract_B` but not outside the derived contracts.

D. internal variable `Cstr` defined in `contract_A` is accessible only inside `contract_A` and not accessible to derived contracts as well.

(b) (5 points) Which of the following statements are true?

```
pragma solidity >=0.8.2 <0.9.0;
contract contract_Q2B {
    uint input;
    mapping(address => uint) bal;
    constructor() public
    {
        input=42;
    }
    function setInput(uint _input) private pure returns(bool){
        input = _input;
        return true;
    }
    //fall back function come here

}

contract sender {
     function setInput(uint _input) public payable {
        address _receiver=0x5B38Da6a701c568545dCfcB03FcB875f56beddC4;
        _receiver.transfer(100);
     }
}
```

A. `function () payable { bal[msg.sender]+=msg.value; }`

B. `function (address _receiver) public payable { bal[msg.sender]+=msg.value; }`

C. `function fallback() () public payable { bal[msg.sender]+=msg.value; }`

D. `function () public payable { bal[msg.sender]+=msg.value; }`

E. Fallback function is executed if contract receives plain Ether without any data.

F. Fallback function is executed if caller calls a function that is not available.

G. If multiple unnamed functions are defined for a contract, the cheapest function is used as fallback function.

(c) (8 points) Complete the following code. What string will return by `retnStr` of `Child` contract?

```
abstract contract ParentA {
    function retnStr() _______________________________ {
        return 'From ParentA';
    }
}
abstract contract ParentB {
    function retnStr() _______________________________ {
        return 'From ParentB';
    }
}
contract Child is ParentA, ParentB {
    function retnStr() _______________________________ {
        return super.returnString();
    }
}
```

(d) (5 points) Which statements about the following contract are true?

```
pragma solidity >=0.8.2 <0.9.0;
abstract contract Book{
    string internal material='papyrus';
    constructor () {}
 }
contract Encyclopedia is Book{
   constructor () {
  }
 function getMaterial() public view returns(string memory){
        return super.material;
   }
}
```

1 - Both the contract `Book` and `Encyclopedia` compiles.

2 - The contract `Book` compiles, but contract `Encyclopedia` does not compile.

3 - The contract contract `Encyclopedia` do not complies because of black constructor as child contract cannot have black constructor.

4 - `getMaterial` returns 'papyrus'.

5 - `super` cannot be used in solidity.

6 - `super` cannot be used for variable name.

A. 1 only

B. 2 only
C. 2 and 5.
D. 2 and 6.
E. 2 and 3.
F. 1 and 4.

Solution:

(e) (5 points) The contract `MathBase` has a modifier to check if the input number is divisible by 3, 5 and 7. Is it possible to override the modifier? If possible, then complete the following code of `MathBase` and `MathDivisor`. If not possible, then complete the following code of `MathBase` only.

```
contract MathBase {
    modifier exactDividedBy2And3(uint _a) ------------------
}
contract MathDivisor is MathBase{
    modifier exactDividedBy2And3(uint _a) ------------------


}
```

(f) (6 points) Will the following code compile? If not, correct the code. Please write the expected event output when the following calls are made:

- `roarCall(-1)`
- `roarCall(0)`
- `roarCall(1)`
- `roarCall(2)`

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

contract BigCat{
    event BigCatRoar(string message);
    function roar(uint level) public pure returns (string memory result){
        require(level >= 0, "Not a valid Roar");
        require(level != 1, "Meow is not Roar");
        emit BigCatRoar("Valid Roar of Big Cat");
        return "Valid Roar of Big Cat";
    }
}
contract Lion is BigCat{
    event LionRoar(string message);
    BigCat public bigCat;
    constructor() {
```

```
        bigCat = new BigCat();
    }
    function roarCall(uint _i) public {
        try bigCat.roar(_i) returns (string memory result) {
            emit LionRoar(result);
        } catch {
            emit LionRoar("Roar call failed");
        }
    }
}
```

Solution:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

contract BigCat{
    event BigCatRoar(string message);









}
contract Lion is BigCat{
    event LionRoar(string message);
    BigCat public bigCat;
```

```
}
```

Question 3 [20 points]

(a) (10 points) "Two contracts are used to raise money for several targets. The first sets the investment threshold rules. The second provides a method to invest tokens for each target." Is the following smart contract safe? If not, what type of vulnerabilities are associated with it?

```
1 contract RuleContract {
2   uint private base_threshold = 50000;
3   function getThreshold ( uint256 currentTarget )
4   public returns ( uint256 ) {
5       require ( currentTarget >= 2020 && currentTarget <= 2050) ;
6       if( currentTarget == 2020) {
7
8           return base_threshold ;
9       }
10      else {
11
12          return currentTarget *100;
13      }
14   }
15 }
16
17 contract InvestContract {
18      uint public total_2030 = 0;
19      uint public cur_invest = 0;
20      // some fixed address for rule contract
21      address fixed_rule = "0 xa214bd6 ..... "
22      function invest (
23          RuleContract rule ,
24          uint target ,
25          uint investToken ) public {
26          assert ( rule == fixed_rule ) ;
27          // the max threshold is 205000;
28          uint threshold = rule . getThreshold ( target ) ;
29          if( threshold == 203000 && threshold <=investToken ) {
30
31           total_2030 += investToken ;
32          cur_invest = investToken ;
33      }
```

```
    else if( investToken > 300000) {

        cur_invest = investToken - threshold /100;
    }
  }
}
```

Solution:

-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-

(b) (10 points) Is the following smart contract safe? If not, what type of vulnerabilities are associated with it?

```
1 contract Trader {
2 TokenSale tokenSale = new TokenSale () ; // Internal Contract defined
    at line in the same file
3 function combination () {
4   tokenSale.buyTokensWithWei () ;
5   tokenSale . buyTokens ( beneficiary ) ;
6   }
7 }
8 contract TokenSale {
9   TokenOnSale tokenOnSale ; // External Contract
10  ...
11  function set ( address _add ) {
12        tokenOnSale = TokenOnSale ( _add ) ;
13  }
14 function buyTokens ( address beneficiary ) {
15      if ( starAllocationToTokenSale > 0) {
16          tokenOnSale.mint ( beneficiary , tokens ) ;
17      }
18  }
19 function buyTokensWithWei () onlyTrader {
20      wallet.transfer ( weiAmount ) ;
21 }
22 }
```

Solution:

-
-
-
-
-
-
-
-
-
-
-
-
-
-
-

-
-
-
-
-
-
-
-
-
-
-
-
-
-
-
-

Question 4 [10 points]

The following Dune query was intended to provide the daily transaction counts from all layer zero protocols, while also displaying a running total of the transactions and a 7-day moving average of the transaction counts, sorted by date.

However, the user forgot to implement some parts in the query, and also there might be other bugs in the query. Your task is to fix these bugs, if any, and also add the missing parts to achieve the desired output from the query.

Here is the query:

```
with lz_send_summary as (
    select date_trunc('day', block_time) as block_date,
        max(block_number) as transaction_count
    from layerzero.send
    group by 1
)

select block_date,
    transaction_count,
    sum(transaction_count) over (order by block_date) as
        accumulate_transaction_count
from lz_send_summary
```

Here is the output that the query produces:

block_date	transaction_count	accumulate_transaction_count
2022-03-15 00:00	33518597	33518597
2022-03-17 00:00	33703086	67221683
2022-03-18 00:00	33793856	101015539
2022-03-19 00:00	33881291	134896830
2022-03-20 00:00	33971151	168867981
2022-03-21 00:00	34061153	202929134
2022-03-22 00:00	34145747	237074881
2022-03-23 00:00	34233734	271308615
2022-03-24 00:00	34317714	305626329
2022-03-25 00:00	34400891	340027220
2022-03-26 00:00	34486610	374513830

636 rows Search... Page 1

And here is what the desired output should look like:

block_date	transaction_count	accumulate_transaction_count	ma_7_day_transaction_count
2022-03-15 00:00	13	13	108446.2
2022-03-17 00:00	592	605	50564.375
2022-03-18 00:00	467	1072	76162.25
2022-03-19 00:00	885	1957	217953.5
2022-03-20 00:00	876	2833	153903.375
2022-03-21 00:00	958	3791	102264.25
2022-03-22 00:00	2348	6139	169165
2022-03-23 00:00	2196	8335	132217.875
2022-03-24 00:00	3259	11594	165528
2022-03-25 00:00	3774	15368	115364.125
2022-03-26 00:00	1974	17342	17515

636 rows Search... Page 1

Hint: Here are the columns of the layerzero.send table:

- blockchain
- source_chain_id
- source_chain_name
- destination_chain_id
- destination_chain_name
- tx_hash
- block_number
- endpoint_contract

- block_time
- trace_address
- adapter_params
- refund_address
- zro_payment_address
- user_address
- transaction_contract
- source_bridge_contract
- destination_bridge_contract
- transfer_type
- currency_symbol
- currency_contract
- amount_usd
- amount_original
- amount_raw

Solution:

-

-

-

-

-

-

-

-

-

-

-

```
-

-

-

-

-


-

-

-
```

Question 5 [21 points]

Write a Contract named String by completing the functions below:

```
contract String{
```

(a) (3 points) Implement the compareTwoString function . It accepts two Strings. If both strings are equals it returns 0, otherwise it returns 1.

```
  function compareTwoString(.............) .................{



 }
```

(b) (3 points) Implement the concatenate function which takes three strings as input and returns the concatenated string.

```
  function concatThreeString(.............).....................{



 }
```

(c) (5 points) Implement the substring function, it takes three input - the string, starting index of the substring and ending index of the substring.

```
function substring(.....................).....................{
```

```
}
```

(d) (5 points) Implement the charAt function which accepts one string and one index value. It returns the char present at index of the string.

```
function charAt(.........................).......................{









}
```

(e) (5 points) Implement the replace function which accepts one string, one letter of the alphabet, and one position value. It will replace the letter at a specific position of the first string.

```
 function replace(.....................) ..................... {









}
```

```
}
```

Question 6 [18 points]

Implement the following Merkle tree with a smart-contract (Solidity) below:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract MerkleTree {

    //         ROOT
    //       /     \
```

```
    //    H1-2       H3-4
    //     /  \       /  \
    //  H1    H2    H3    H4
    //   |     |     |     |
    //  TX1   TX2   TX3   TX4
    bytes32[] public hashes;
    string[4] transactions = [
        "TX1: Sherlock -> John",
        "TX2: John -> Sherlock",
        "TX3: John -> Mary",
        "TX4: Mary -> Sherlock"
    ];
```

(a) (6 points) Constructor.

```
    constructor() {



    }
```

(b) (6 points) Implement verify function to verify transaction. If one put the values transaction, index, root and proof into the verify function call, it will return true. If there is even the slightest change in any data we get false.

```
    function verify(string memory transaction, uint index, bytes32 root,
        bytes32[] memory proof) public pure returns(bool) {



    }
```

(c) (6 points) Implement makeHash which returns the hash value of an transaction.

```
    function makeHash(string memory input) public pure returns(bytes32) {



    }
```

```
}
```

Solution:

-

Question 7 [14 points]

(a) Write the following contract using solidity.

```
# EVENTS:
NewSolutionFound: event({_addressOfWinner: indexed(address), _solution:
    uint256})
BountyTransferred: event({_to: indexed(address), _amount: wei_value})
```

```
BountyIncreased: event({_amount: wei_value})
CompetitionTimeExtended: event({_to: uint256})

# STATE VARIABLES:
owner: public(address)

x1: public(uint256)
x2: public(uint256)

bestSolution: public(uint256)
addressOfWinner: public(address)

durationInBlocks: public(uint256)
competitionEnd: public(uint256)
claimPeriodeLength: public(uint256)


# METHODS:
@public
def __init__(_durationInBlocks: uint256):
    self.owner = msg.sender
    self.bestSolution = 0
    self.durationInBlocks = _durationInBlocks
    self.competitionEnd = block.number + _durationInBlocks
    self.addressOfWinner = ZERO_ADDRESS
    # set claim periode to three days
    # assuming an average blocktime of 14 seconds -> 86400/14
    self.claimPeriodeLength = 6172


@public
@payable
def __default__():
    # return any funds sent to the contract address directly
    send(msg.sender, msg.value)


@private
def _calculateNewSolution(_x1: uint256, _x2: uint256) -> uint256:
    # check new parameters against constraints
    assert _x1 <= 40
    assert _x2 <= 35
    assert (3 * _x1) + (2 * _x2) <= 200
    assert _x1 + _x2 <= 120
    assert _x1 > 0 and _x2 > 0
    # calculate and return new solution
    return (4 * _x1) + (6 * _x2)
```

```
@public
def submitSolution(_x1: uint256, _x2: uint256) -> uint256:
    newSolution: uint256
    newSolution = self._calculateNewSolution(_x1, _x2)
    assert newSolution > self.bestSolution
    # save the solution and its values
    self.x1 = _x1
    self.x2 = _x2
    self.bestSolution = newSolution
    self.addressOfWinner = msg.sender
    log.NewSolutionFound(msg.sender, newSolution)
    return newSolution


@public
def claimBounty():
    assert block.number > self.competitionEnd
    if (self.addressOfWinner == ZERO_ADDRESS):
        # no solution was found -> extend duration of competition
        self.competitionEnd = block.number + self.durationInBlocks
    else:
        assert block.number < (self.competitionEnd +
            self.claimPeriodeLength)
        assert msg.sender == self.addressOfWinner
        send(self.addressOfWinner, self.balance)
        # extend duration of competition
        self.competitionEnd = block.number + self.durationInBlocks
        log.BountyTransferred(self.addressOfWinner, self.balance)


@public
@payable
def topUpBounty():
    log.BountyIncreased(msg.value)


@public
def extendCompetition():
    # only if no valid solution has been submitted
    assert self.addressOfWinner == ZERO_ADDRESS
    assert block.number > (self.competitionEnd + self.claimPeriodeLength)
    # extend duration of competition
    self.competitionEnd = block.number + self.durationInBlocks
    # reset winner address
    self.addressOfWinner = ZERO_ADDRESS
```

```
    log.CompetitionTimeExtended(self.competitionEnd)
```

Solution:

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-

-