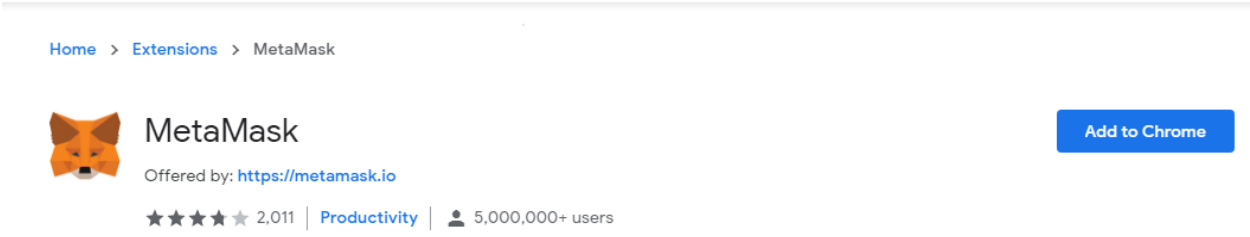
BLOCK TECHNOLOGY

Assignment 1:

Step 1: Go to [Chrome Web Store Extensions Section](#).

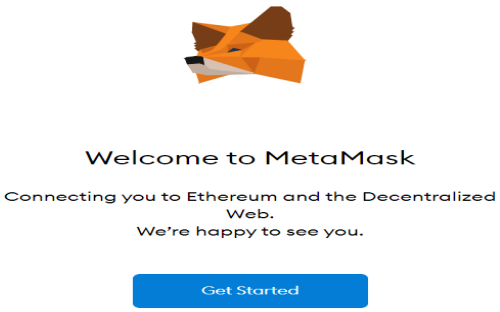
Step 2: Search *MetaMask.*

Step 3: Check the number of downloads to make sure that the legitimate MetaMask is being installed, as hackers might try to make clones of it.

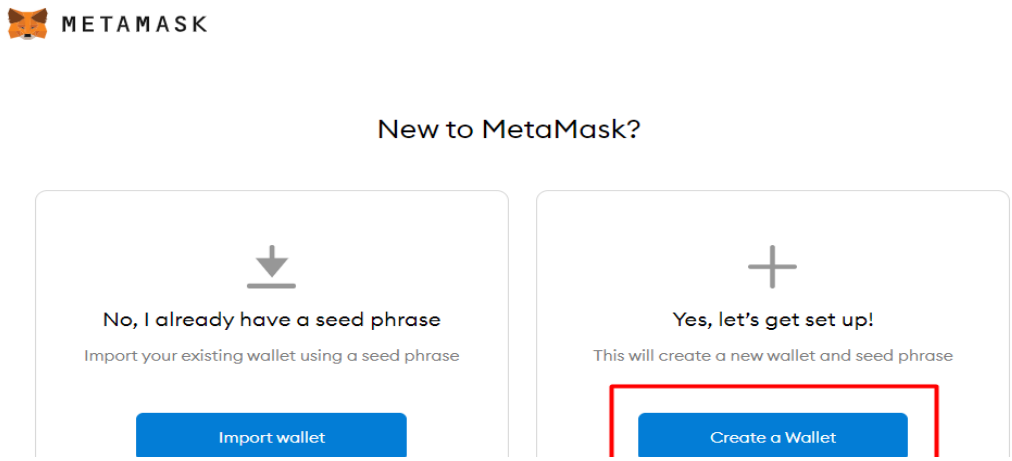


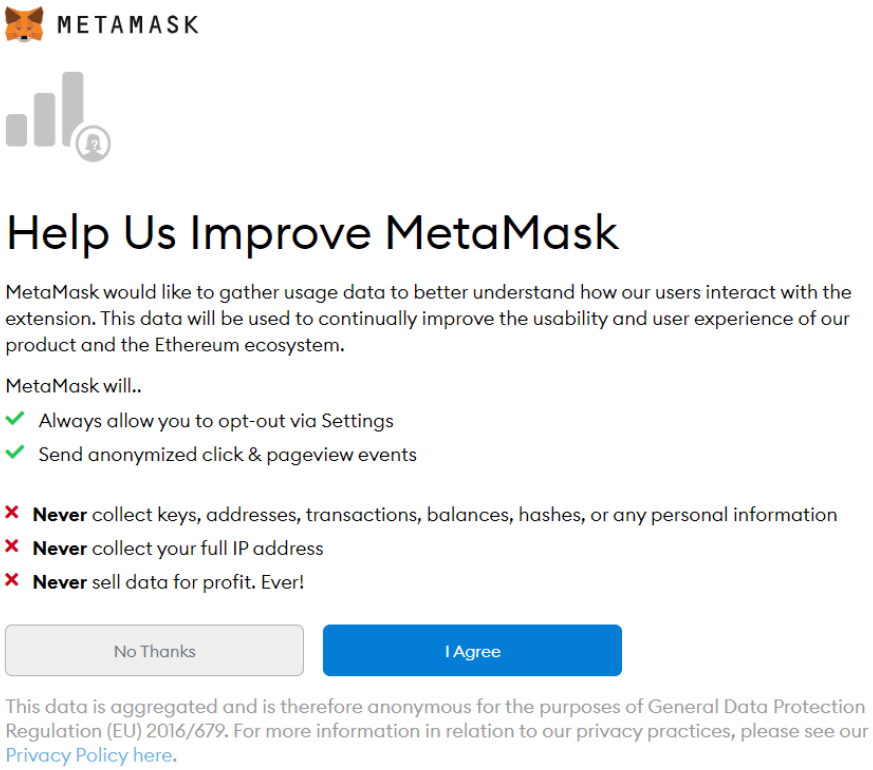
Step 4: Click the *Add to Chrome* button.

Step 5: Once installation is complete this page will be displayed. Click on the *Get Started* button.

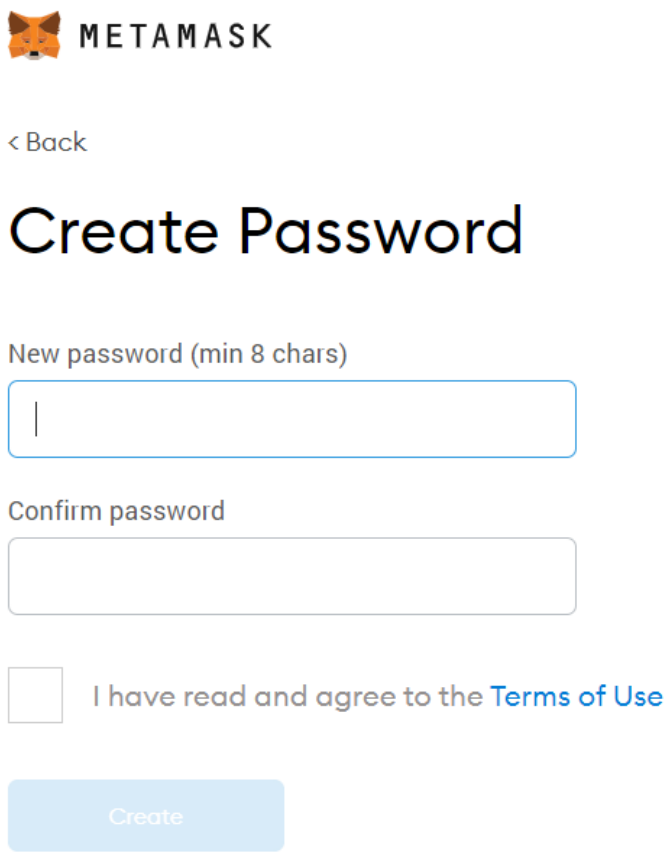


Step 6: This is the first time creating a wallet, so click the *Create a Wallet* button. If there is already a wallet then import the already created using the *Import Wallet* button.

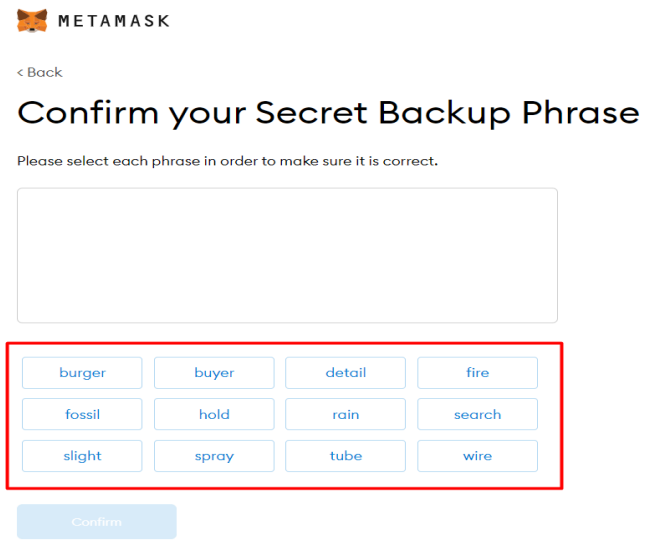




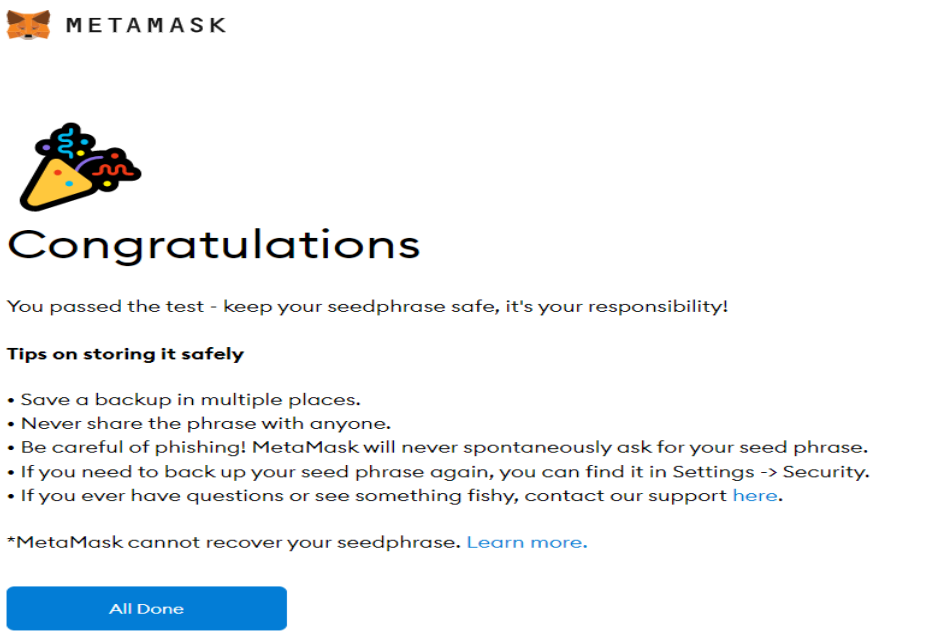
Step 7: Create a password for your wallet. This password is to be entered every time the browser is launched and wants to use MetaMask. A new password needs to be created if chrome is uninstalled or if there is a switching of browsers. In that case, go through the *Import Wallet* button. This is because MetaMask stores the keys in the browser. Agree to *Terms of Use*.

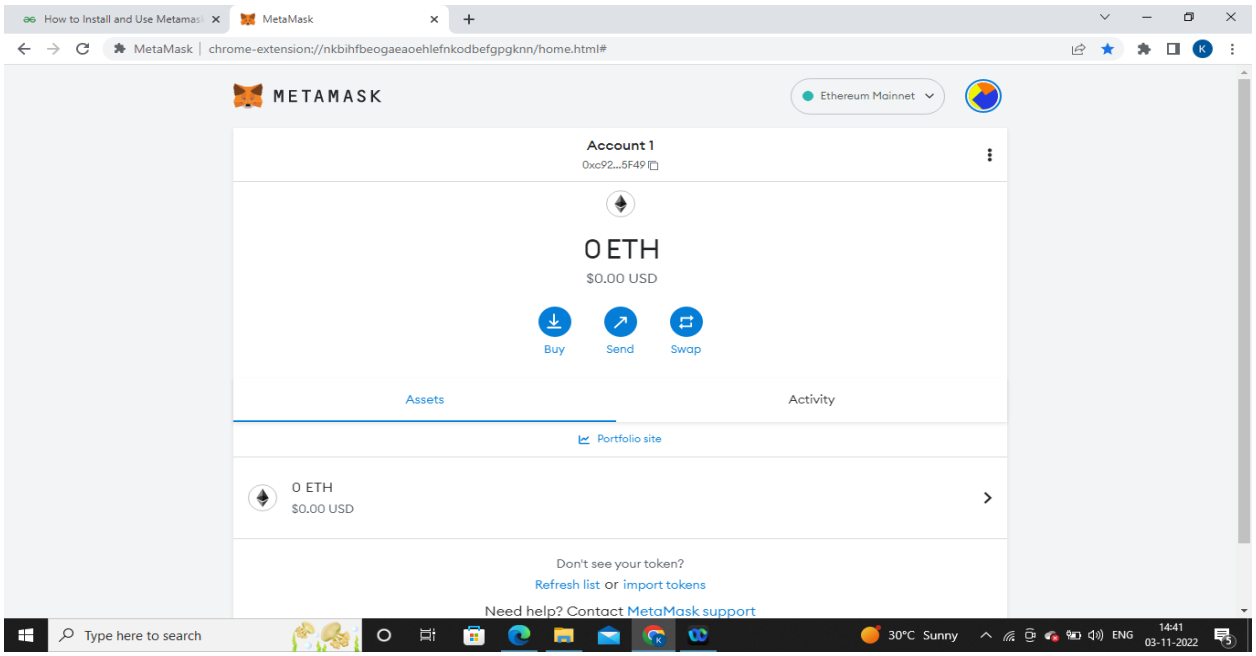


Step 8: Click the buttons respective to the order of the words in your seed phrase. In other words, type the seed phrase using the button on the screen. If done correctly the *Confirm* button should turn blue.

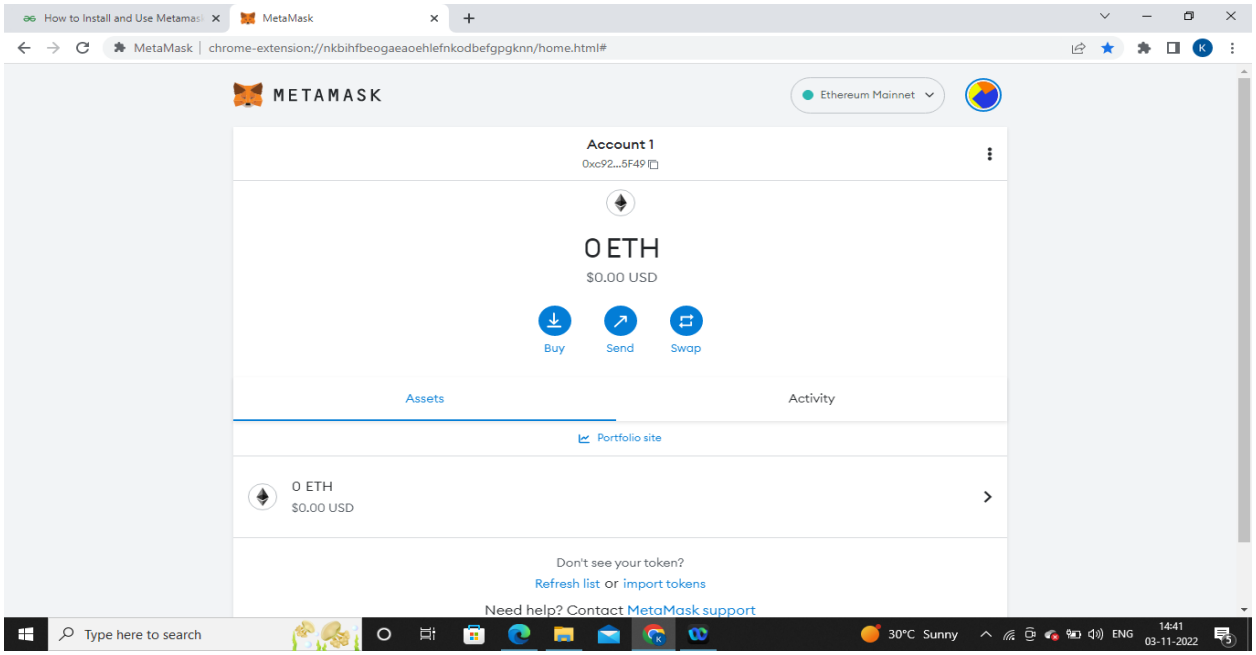


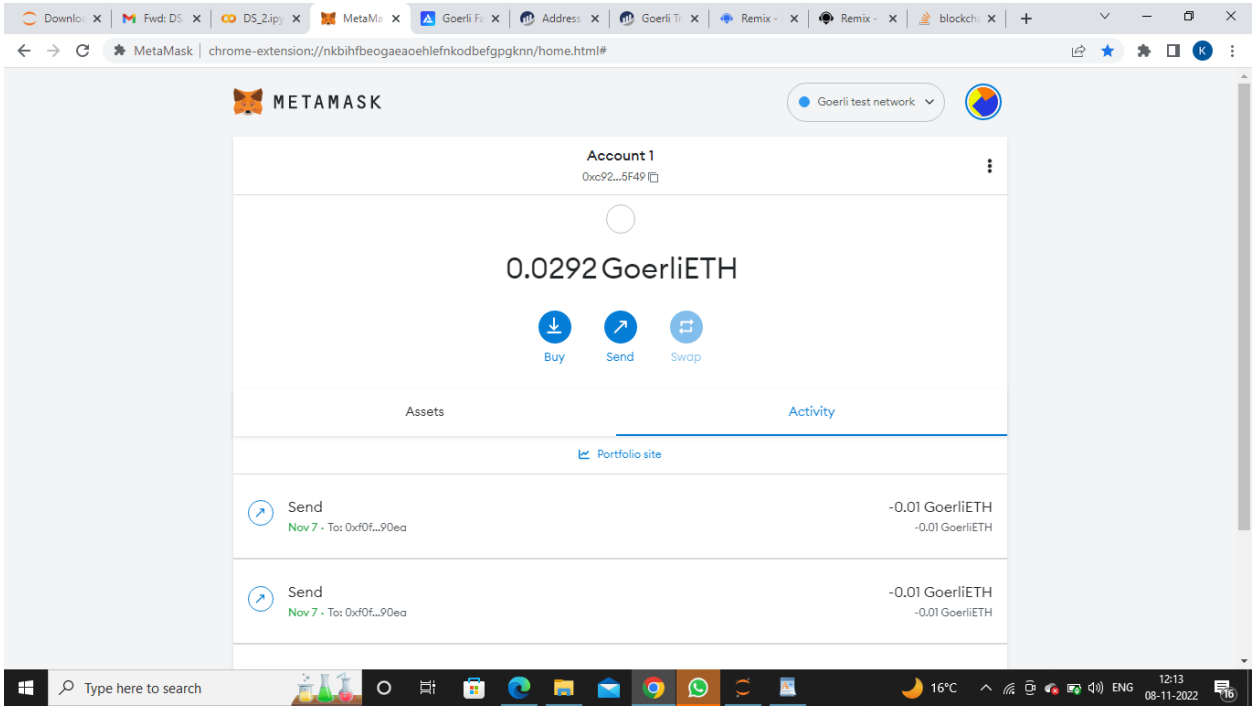
Click the *Confirm* button. Please follow the tips mentioned.





Assignment 2:





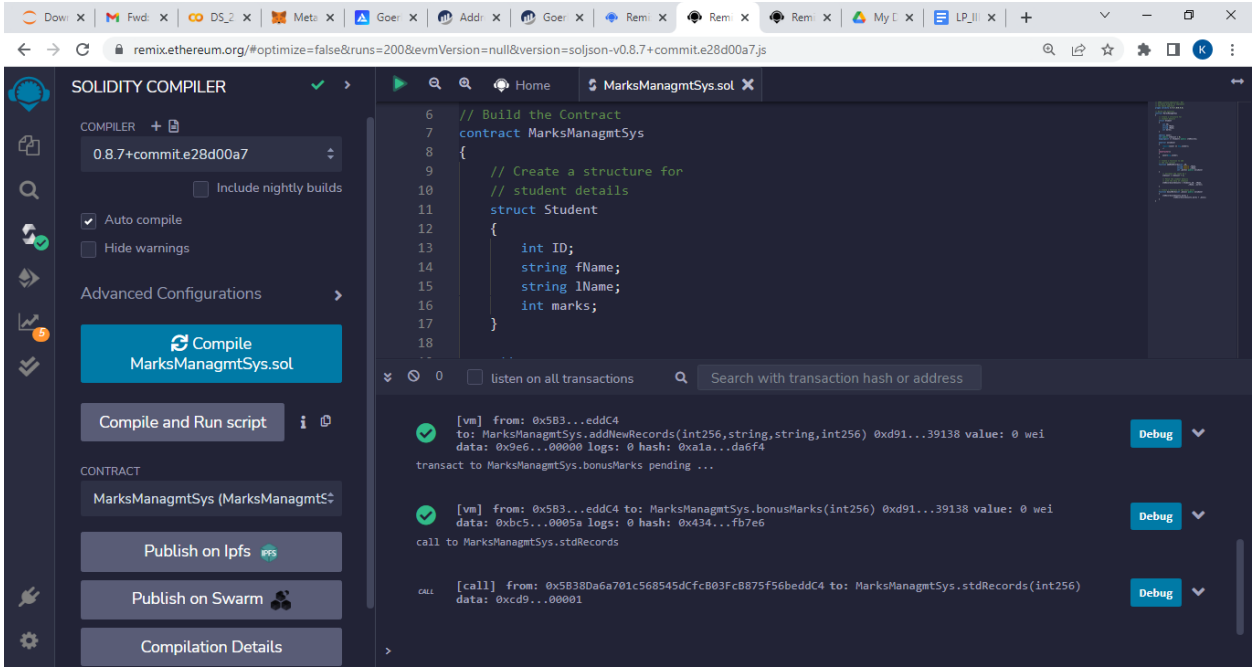
Assignment no 3:

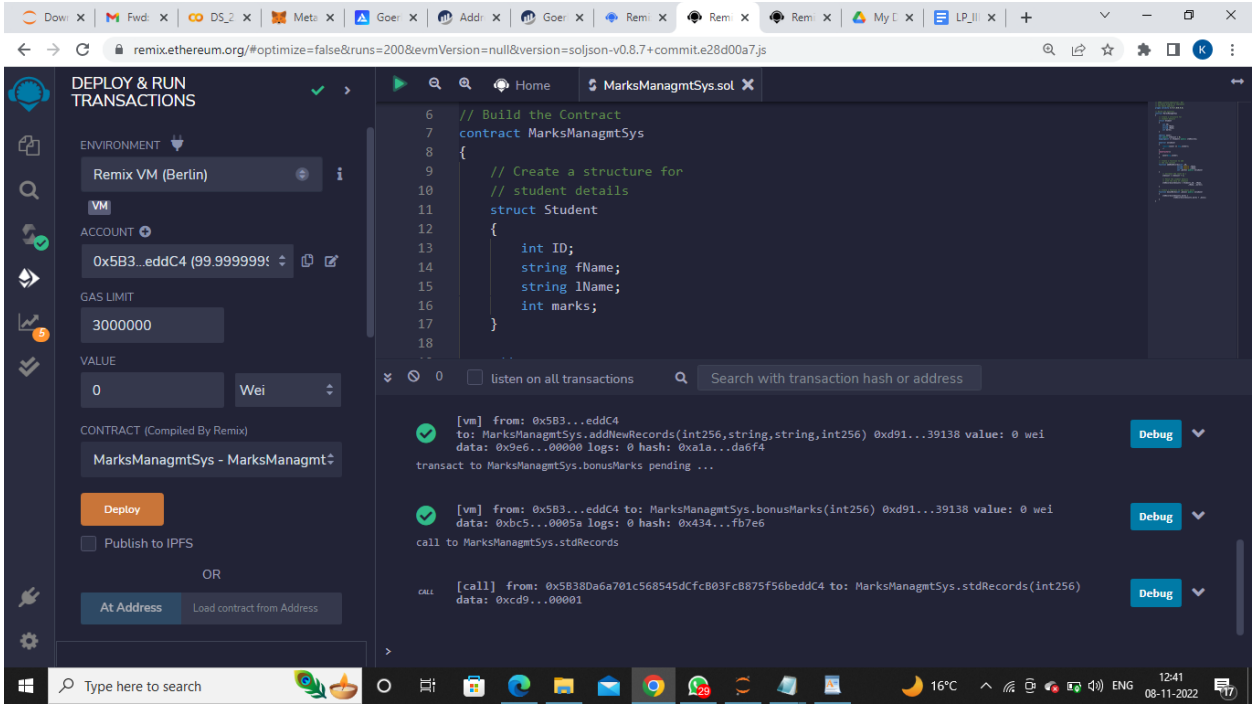
Input:

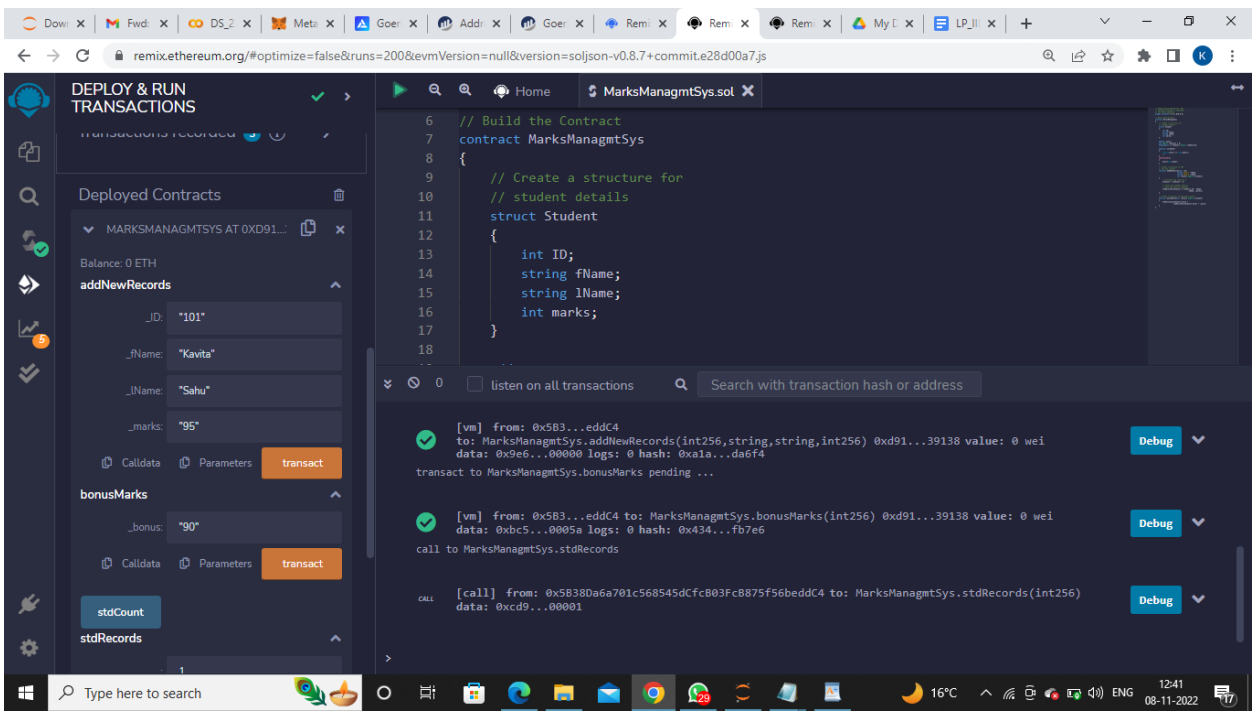
```
// SPDX-License-Identifier: MIT
// Solidity program to implement
// the above approach
pragma solidity >= 0.7.0<0.9.0;

// Build the Contract
contract MarksManagmtSys
{
	// Create a structure for
	// student details
	struct Student
	{
		int ID;
		string fName;
		string lName;
		int marks;
	}

	address owner;
	int public stdCount = 0;
	mapping(int => Student) public stdRecords;

	modifier onlyOwner
	{
		require(owner == msg.sender);
		_;
	}
	constructor()
	{
		owner=msg.sender;
	}

	// Create a function to add
	// the new records
	function addNewRecords(int _ID,
						string memory _fName,
						string memory _lName,
						int _marks) public onlyOwner
	{
		// Increase the count by 1
```

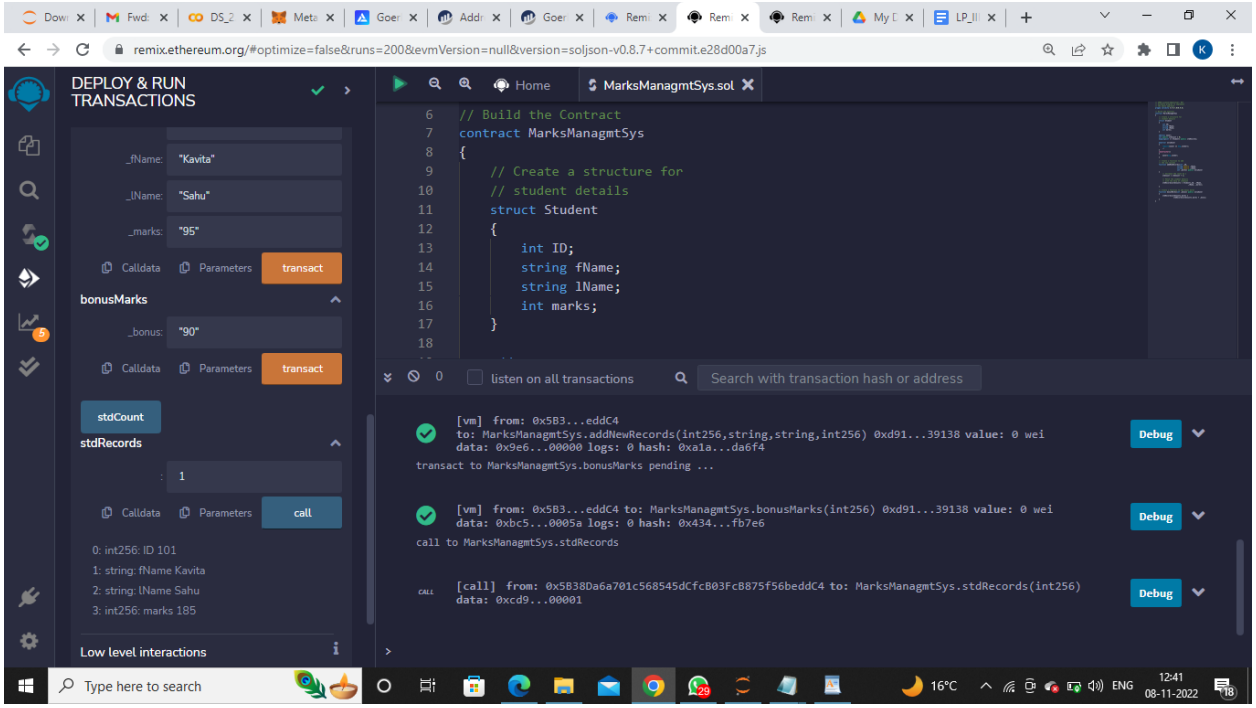
```
        stdCount = stdCount + 1;

        // Fetch the student details
        // with the help of stdCount
        stdRecords[stdCount] = Student(_ID, _fName,
                                                    _lName, _marks);
    }

    // Create a function to add bonus marks
    function bonusMarks(int _bonus) public onlyOwner
    {
        stdRecords[stdCount].marks =
                        stdRecords[stdCount].marks + _bonus;
    }
}
```

Output:




remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js
DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
Remix VM (Berlin)
VM
ACCOUNT
0x5B3...eddC4 (99.999999
GAS LIMIT
3000000
VALUE
0
Wei
CONTRACT (Compiled By Remix)
MarksManagmtSys - MarksManagmt
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Home
MarksManagmtSys.sol
// Build the Contract
contract MarksManagmtSys
{
// Create a structure for
// student details
struct Student
{
int ID;
string fName;
string lName;
int marks;
}
listen on all transactions
Search with transaction hash or address
[vm] from: 0x5B3...eddC4
to: MarksManagmtSys.addNewRecords(int256,string,string,int256) 0xd91...39138 value: 0 wei
data: 0x9e6...00000 logs: 0 hash: 0xa1a...da6f4
transact to MarksManagmtSys.bonusMarks pending ...
Debug
[vm] from: 0x5B3...eddC4 to: MarksManagmtSys.bonusMarks(int256) 0xd91...39138 value: 0 wei
data: 0xbc5...0005a logs: 0 hash: 0x434...fb7e6
call to MarksManagmtSys.stdRecords
Debug
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: MarksManagmtSys.stdRecords(int256)
data: 0xcd9...00001
Debug
Type here to search
16°C
ENG
12:41
08-11-2022


remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js
DEPLOY & RUN TRANSACTIONS
Deployed Contracts
MARKSMANAGMTSYS AT 0XD91...
Balance: 0 ETH
addNewRecords
_ID: "101"
_fName: "Kavita"
_lName: "Sahu"
_marks: "95"
Calldata
Parameters
transact
bonusMarks
_bonus: "90"
Calldata
Parameters
transact
stdCount
stdRecords
Home
MarksManagmtSys.sol
// Build the Contract
contract MarksManagmtSys
{
// Create a structure for
// student details
struct Student
{
int ID;
string fName;
string lName;
int marks;
}
listen on all transactions
Search with transaction hash or address
[vm] from: 0x5B3...eddC4
to: MarksManagmtSys.addNewRecords(int256,string,string,int256) 0xd91...39138 value: 0 wei
data: 0x9e6...00000 logs: 0 hash: 0xa1a...da6f4
transact to MarksManagmtSys.bonusMarks pending ...
Debug
[vm] from: 0x5B3...eddC4 to: MarksManagmtSys.bonusMarks(int256) 0xd91...39138 value: 0 wei
data: 0xbc5...0005a logs: 0 hash: 0x434...fb7e6
call to MarksManagmtSys.stdRecords
Debug
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: MarksManagmtSys.stdRecords(int256)
data: 0xcd9...00001
Debug
Type here to search
16°C
ENG
12:41
08-11-2022


remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js
DEPLOY & RUN TRANSACTIONS
_fName: "Kavita"
_lName: "Sahu"
_marks: "95"
Calldata
Parameters
transact
bonusMarks
_bonus: "90"
Calldata
Parameters
transact
stdCount
stdRecords
: 1
Calldata
Parameters
call
0: int256: ID 101
1: string: fName Kavita
2: string: lName Sahu
3: int256: marks 185
Low level interactions
Home
MarksManagmtSys.sol
// Build the Contract
contract MarksManagmtSys
{
// Create a structure for
// student details
struct Student
{
int ID;
string fName;
string lName;
int marks;
}
listen on all transactions
Search with transaction hash or address
[vm] from: 0x5B3...eddC4
to: MarksManagmtSys.addNewRecords(int256,string,string,int256) 0xd91...39138 value: 0 wei
data: 0x9e6...00000 logs: 0 hash: 0xa1a...da6f4
transact to MarksManagmtSys.bonusMarks pending ...
Debug
[vm] from: 0x5B3...eddC4 to: MarksManagmtSys.bonusMarks(int256) 0xd91...39138 value: 0 wei
data: 0xbc5...0005a logs: 0 hash: 0x434...fb7e6
call to MarksManagmtSys.stdRecords
Debug
CALL
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: MarksManagmtSys.stdRecords(int256)
data: 0xcd9...00001
Debug
Type here to search
16°C
ENG
12:41
08-11-2022

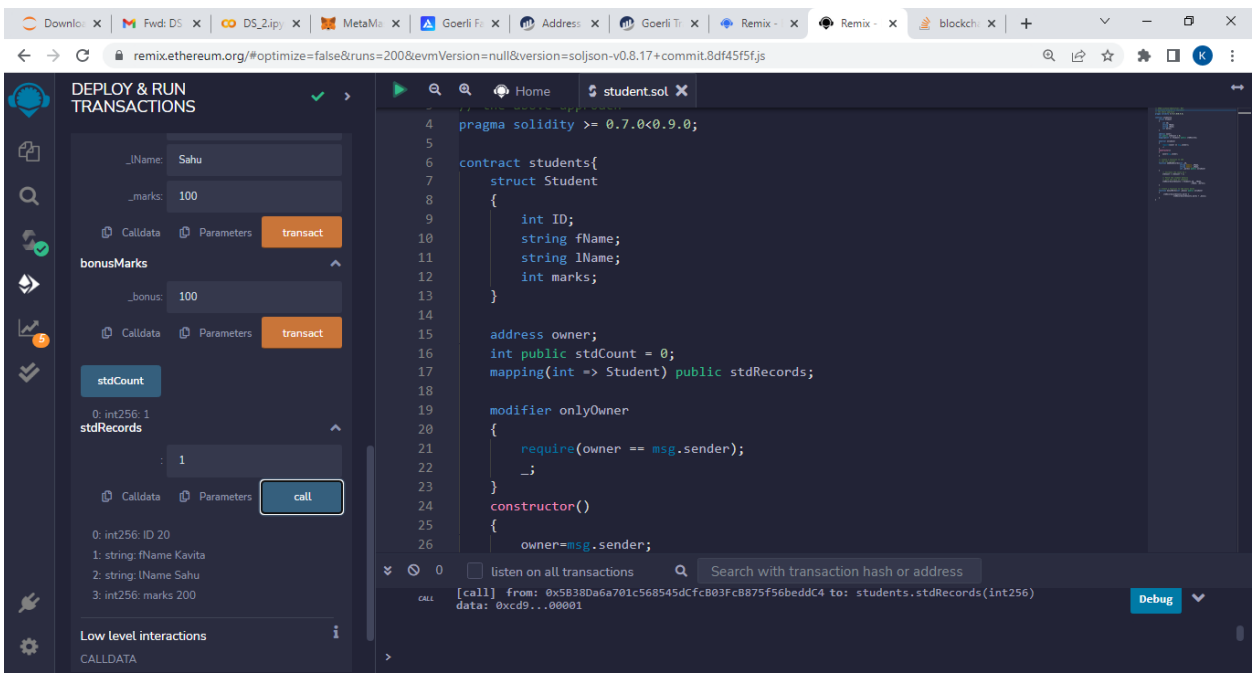
Assignment 4:

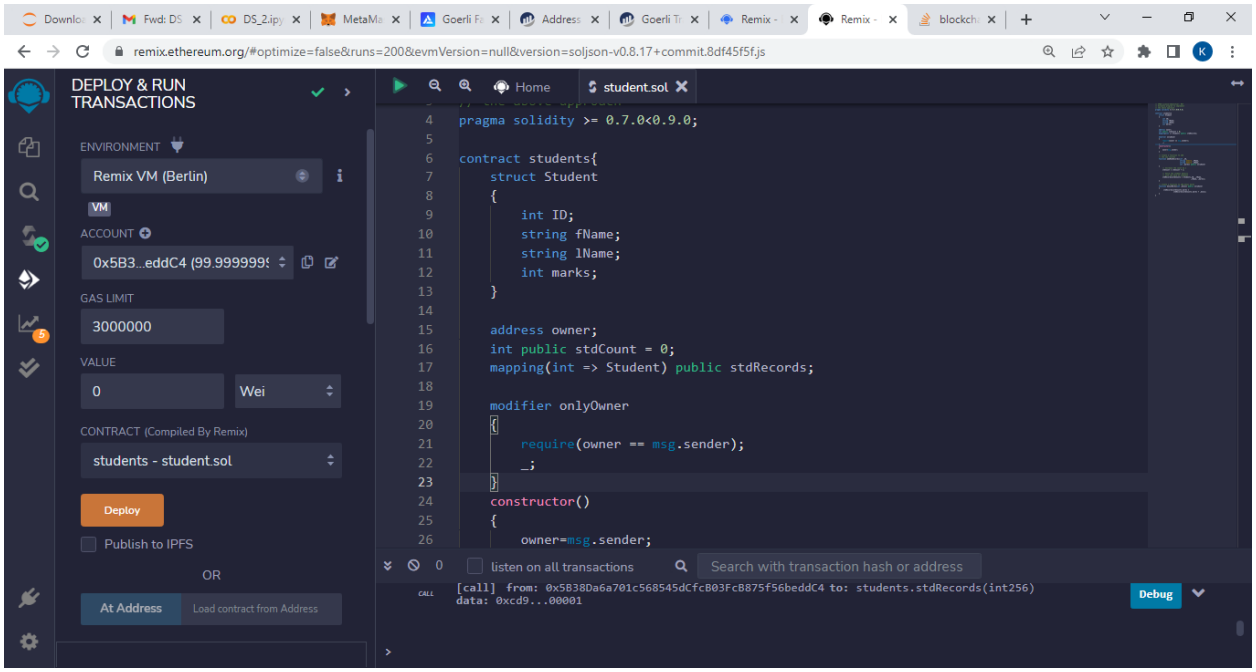
Input:

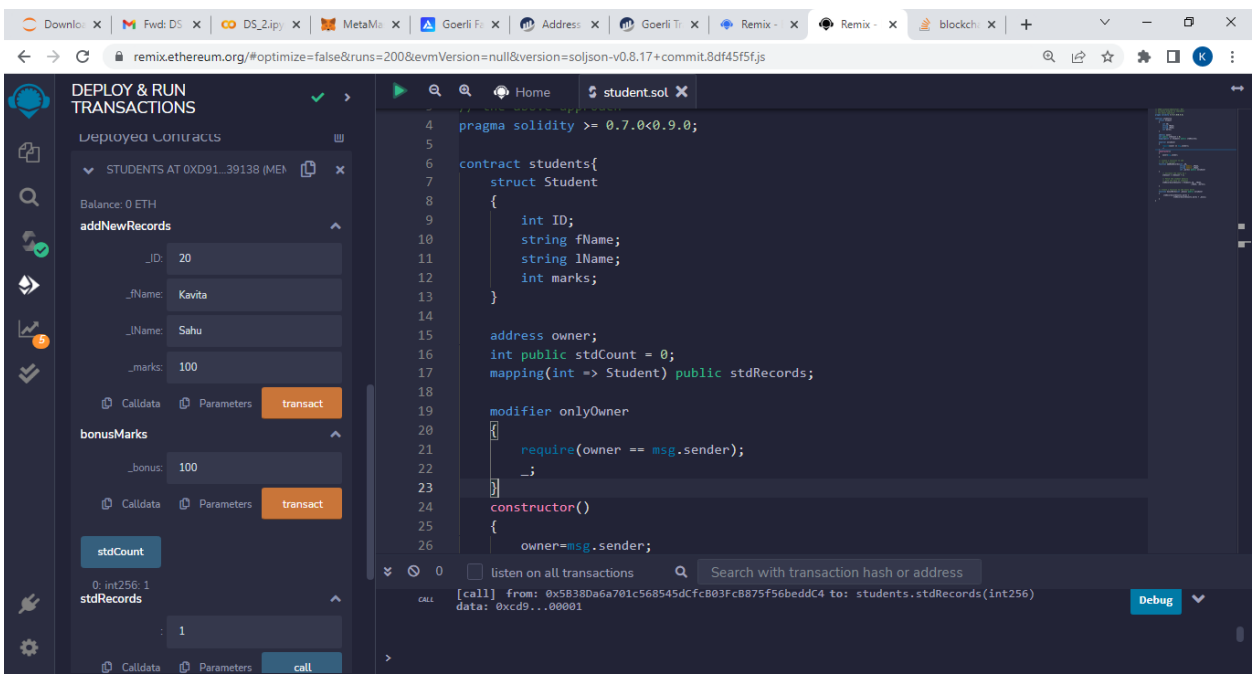
```
// SPDX-License-Identifier: MIT
// Solidity program to implement
// the above approach
pragma solidity >= 0.7.0<0.9.0;

contract students{
    struct Student
    {
        int ID;
        string fName;
        string lName;
        int marks;
    }

    address owner;
    int public stdCount = 0;
    mapping(int => Student) public stdRecords;

    modifier onlyOwner
    {
        require(owner == msg.sender);
        _;
    }
    constructor()
    {
        owner=msg.sender;
    }

    // Create a function to add
    // the new records
    function addNewRecords(int _ID,
                           string memory _fName,
                           string memory _lName,
                           int _marks) public onlyOwner
    {
        // Increase the count by 1
        stdCount = stdCount + 1;
```

```
        // Fetch the student details
        // with the help of stdCount
        stdRecords[stdCount] = Student(_ID, _fName,
                                _lName, _marks);
    }

    // Create a function to add bonus marks
    function bonusMarks(int _bonus) public onlyOwner
    {
        stdRecords[stdCount].marks =
                stdRecords[stdCount].marks + _bonus;
    }
}
```

Output:




remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.17+commit.8df45f5f.js
DEPLOY & RUN TRANSACTIONS
ENVIRONMENT
Remix VM (Berlin)
VM
ACCOUNT
0x5B3...eddC4 (99.999999
GAS LIMIT
3000000
VALUE
0
Wei
CONTRACT (Compiled By Remix)
students - student.sol
Deploy
Publish to IPFS
OR
At Address
Load contract from Address
Home
student.sol
pragma solidity >= 0.7.0<0.9.0;
contract students{
struct Student
{
int ID;
string fName;
string lName;
int marks;
}
address owner;
int public stdCount = 0;
mapping(int => Student) public stdRecords;
modifier onlyOwner
{
require(owner == msg.sender);
_;
}
constructor()
{
owner=msg.sender;
listen on all transactions
Search with transaction hash or address
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: students.stdRecords(int256)
data: 0xcd9...00001
Debug


remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.17+commit.8df45f5f.js
DEPLOY & RUN TRANSACTIONS
Deployed Contracts
STUDENTS AT 0XD91...39138 (MEM
Balance: 0 ETH
addNewRecords
_ID: 20
_fName: Kavita
_lName: Sahu
_marks: 100
Calldata
Parameters
transact
bonusMarks
_bonus: 100
Calldata
Parameters
transact
stdCount
0: int256: 1
stdRecords
: 1
Calldata
Parameters
call
Home
student.sol
pragma solidity >= 0.7.0<0.9.0;
contract students{
struct Student
{
int ID;
string fName;
string lName;
int marks;
}
address owner;
int public stdCount = 0;
mapping(int => Student) public stdRecords;
modifier onlyOwner
{
require(owner == msg.sender);
_;
}
constructor()
{
owner=msg.sender;
listen on all transactions
Search with transaction hash or address
[call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: students.stdRecords(int256)
data: 0xcd9...00001
Debug

Mini project:

Input:

```
// SPDX-License-Identifier: MIT
// Solidity program to implement
// the above approach
pragma solidity >= 0.7.0<0.9.0;

contract ElectionFact {

 struct ElectionDet {
 address deployedAddress;
 string el_n;
 string el_d;
 }

 mapping(string=>ElectionDet) companyEmail;

 function createElection(string memory email,string memory election_name, string
memory election_description) public{
 address newElection = new Election(msg.sender , election_name,
election_description);

 companyEmail[email].deployedAddress = newElection;
 companyEmail[email].el_n = election_name;
 companyEmail[email].el_d = election_description;
 }

 function getDeployedElection(string memory email) public view returns
(address,string,string) {
 address val = companyEmail[email].deployedAddress;
 if(val == 0)
 return (0,"","Create an election.");
 else
 return
(companyEmail[email].deployedAddress,companyEmail[email].el_n,companyEmail[em
ail].el_d);
 }
}
contract Election {
 //election_authority's address
 address election_authority;
```

```
string election_name;
string election_description;
bool status;

//election_authority's address taken when it deploys the contract
constructor(address authority , string name, string description) public {
election_authority = authority;
election_name = name;
election_description = description;
status = true;
}
//Only election_authority can call this function
modifier owner() {
require(msg.sender == election_authority, "Error: Access Denied.");
_;
}
//candidate election_description
struct Candidate {
string candidate_name;
string candidate_description;
string imgHash;
uint8 voteCount;
string email;
}
//candidate mapping
mapping(uint8=>Candidate) public candidates;
//voter election_description
struct Voter {
uint8 candidate_id_voted;
bool voted;
}
//voter mapping
mapping(string=>Voter) voters;
//counter of number of candidates
uint8 numCandidates;
//counter of number of voters
uint8 numVoters;
//function to add candidate to mapping
function addCandidate(string memory candidate_name, string memory
candidate_description, string memory imgHash,string memory email) public owner {
uint8 candidateID = numCandidates++; //assign id of the candidate
candidates[candidateID] =
Candidate(candidate_name,candidate_description,imgHash,0,email); //add the values to
the mapping
```

```
 }
 //function to vote and check for double voting
 function vote(uint8 candidateID,string e) public {
 //if false the vote will be registered
 require(!voters[e].voted, "Error:You cannot double vote");

 voters[e] = Voter (candidateID,true); //add the values to the mapping
 numVoters++;
 candidates[candidateID].voteCount++; //increment vote counter of candidate

 }
 //function to get count of candidates
 function getNumOfCandidates() public view returns(uint8) {
 return numCandidates;
 }
 //function to get count of voters
 function getNumOfVoters() public view returns(uint8) {
 return numVoters;
 }
 //function to get candidate information
 function getCandidate(uint8 candidateID) public view returns (string memory, string
memory, string memory, uint8,string memory) {
 return (candidates[candidateID].candidate_name,
candidates[candidateID].candidate_description, candidates[candidateID].imgHash,
candidates[candidateID].voteCount, candidates[candidateID].email);
 }
 //function to return winner candidate information
 function winnerCandidate() public view owner returns (uint8) {
 uint8 largestVotes = candidates[0].voteCount;
 uint8 candidateID;
 for(uint8 i = 1;i<numCandidates;i++) {
 if(largestVotes < candidates[i].voteCount) {
 largestVotes = candidates[i].voteCount;
 candidateID = i;
 }
 }
 return (candidateID);
 }

 function getElectionDetails() public view returns(string, string) {
 return (election_name,election_description);
 }
}
```

Output: