**SET 1**

**1)Install Ganache(Personal block chain) and metamask (Show the installation Steps) . Compile and deploy a election smart contract in the personal blockchain using injected web 3 environment(Metamask wallet) .**

**CODE: -**

pragma solidity ^0.4.2;

contract Election {

// Model a Candidate

struct Candidate {

uint id;

string name;

uint voteCount;

}

// Store accounts that have voted

mapping(address => bool) public voters;

// Store Candidates

// Fetch Candidate

mapping(uint => Candidate) public candidates;

// Store Candidates Count

uint public candidatesCount;

// voted event

event votedEvent (

uint indexed \_candidateId

);

constructor() public {

addCandidate("N MODI, BJP");

addCandidate("A kejriwal, AAP");

addCandidate("Rahul G, Congress");

addCandidate("Nikhil, JDS");

}

function addCandidate (string \_name) private {

candidatesCount ++;

candidates[candidatesCount] = Candidate(candidatesCount, \_name, 0);

}

function vote (uint \_candidateId) public {

// require that they haven't voted before

require(!voters[msg.sender]);

// require a valid candidate

require(\_candidateId > 0 && \_candidateId <= candidatesCount);

// record that voter has voted

voters[msg.sender] = true;

// update candidate vote Count

candidates[\_candidateId].voteCount ++;

// trigger voted event emit

emit votedEvent(\_candidateId);

}

}

**2)Write a program to implement of Caesar Cipher (Symmetric Encryption) in python / C++ /js**

**CODE: -**

def encrypt(text, s):

result = ""

for i in range(len(text)):

char = text[i]

if (char.isupper()):

result += chr((ord(char) + shift - 65) % 26 + 65)

else:

result += chr((ord(char) + shift - 97) % 26 + 97)

return result

def decrypt(text, s):

result = ""

for i in range(len(text)):

char = text[i]

if (char.isupper()):

result += chr((ord(char) + s - 64) % 26 + 65)

else:

result += chr((ord(char) + s - 97) % 26 + 97)

return result

shift=7

s = 26-shift

text = "ATTACK"

text2=encrypt(text,shift)

print("Text : " + text)

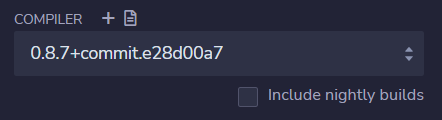
print("Shift : " + str(shift))

print("Cipher: " + encrypt(text, shift))

print("DeCipher: " + decrypt(text2, shift))

**SET 2**

**1)Use Remix online ide to compile and deploy the Bank account smart contract. Execute the smart contract and show the output.**

****

**CODE: -**

pragma solidity 0.8.7;

contract Bank {

address public owner;

uint public balance;

constructor() {

owner = msg.sender;

}

function deposit() external payable {

balance += msg.value;

}

function withdraw(uint \_amount) external {

require(balance >= \_amount, "Insufficient balance!");

payable(owner).transfer(\_amount);

balance -= \_amount;

}

}

**2) Create a structure Consumer with Name , Address, Consumer ID, Units and Amount as**

**members.Write a program in solidity to calculate the total electricity bill according to the given condition:**

**For first 50 units Rs. 0.50/unit**

**For next 100 units Rs. 0.75/unit**

**For next 100 units Rs. 1.20/unit**

**For unit above 250 Rs. 1.50/unit**

**An additional surcharge of 20% is added to the bill. Display the information of 5 such consumers along with their units consumed and amount.**

**CODE: -**

pragma solidity ^0.5.0;

pragma experimental ABIEncoderV2;

contract q4 {

struct Consumer {

uint256 units;

string name;

string addr;

uint256 consumerID;

uint256 amount;

}

Consumer[] consumer;

function addConsumerInfo(

uint256 units,

string memory name,

string memory addr,

uint256 consumerID

) public {

consumer.push(Consumer(units, name, addr, consumerID, 0));

}

function getBillAmount() public {

uint256 i = 0;

uint256 amount = 0;

uint256 surcharge = 0;

uint256 units = 0;

for (i = 0; i < consumer.length; i++) {

units = consumer[i].units;

if (units <= 50) {

amount = ((units \* 1) / 2);

} else if (units <= 150) {

amount = 25 + (((units - 50) \* 3) / 4);

} else if (units <= 250) {

amount = 100 + (((units - 150) \* 6) / 5);

} else {

amount = 220 + (((units - 250) \* 3) / 2);

}

surcharge = (amount \* 20) / 100;

amount = amount + surcharge;

consumer[i].amount = amount;

}

}

function displayConsumerInfo(uint256 consumerID)

public

view

returns (

uint256,

string memory,

string memory,

uint256,

uint256

)

{

uint256 i = 0;

for (i = 0; i < consumer.length; i++) {

if (consumer[i].consumerID == consumerID) {

return (

consumer[i].units,

consumer[i].name,

consumer[i].addr,

consumer[i].consumerID,

consumer[i].amount

);

}

}

return (

consumer[0].units,

consumer[0].name,

consumer[0].addr,

consumer[0].consumerID,

consumer[0].amount

);

}

function displayAllInfo() public view returns (Consumer[] memory) {

return consumer;

}

}

**SET 5**

**Q.1) Write a program in solidity to calculate the tax with the given following conditions.**

**1. if income is less than, 1,50,000 then no tax**

**2. if taxable income is in the range 1,50,001-300,000 then charge 10% tax**

**3. if taxable income is in the range 3,00,001-500,000 then charge 20% tax**

**4. if taxable income is above 5,00,001 then charge 30% tax.**

**CODE**

pragma solidity ^0.5.0;

contract q2 {

function cal\_tax(uint256 income) public view returns (uint256) {

if(income<150000)

{

return 0;

}

else if((income>=150000) && (income<300000))

{

uint256 tax = income \* 10/100;

return tax;

}

else if((income>=300001) && (income<500000))

{

uint256 tax = income \* 20/100;

return tax;

}

else if(income>=500001)

{

uint256 tax = income \* 30/100;

return tax;

}

}

}

**SET 6**

**1) Write a program in solidity to create a structured student with Roll no, Name,Class,**

**Department,Course enrolled as variables.**

**i) Add information about 5 students.**

**ii)Search for a student using Roll no**

**iii)Display all information**

**CODE: -**

pragma solidity ^0.5.0;

pragma experimental ABIEncoderV2;

contract q3 {

struct student {

uint256 rollNo;

string name;

string placedCompany;

string degreeCourse;

}

student[] s;

function addStudent(

uint256 rollNo,

string memory name,

string memory placedCompany,

string memory degreeCourse

) public {

s.push(student(rollNo, name, placedCompany, degreeCourse));

}

function getStudentByRollNo(uint256 rollNumber)

public

view

returns (uint256, string memory)

{

uint256 i = 0;

for (i = 0; i < s.length; i++) {

if (s[i].rollNo == rollNumber) {

return (s[i].rollNo, s[i].name);

}

}

return (s[0].rollNo, s[0].name);

}

function getAllStudents() public view returns (student[] memory) {

return s;}}

**SET 7**

**Dear you have to solve two question 15 marks each**

**1)Write a program in solidity to input the basic salary of an employee and calculate gross salary**

**according to given conditions.**

**Basic Salary <= 10000 : HRA = 20%, DA = 80%**

**Basic Salary is between 10001 to 20000 : HRA = 25%, DA = 90%**

**Basic Salary >= 20001 : HRA = 30%, DA = 95%**

**CODE: -**

pragma solidity ^0.5.0;

//pragma experimental ABIEncoderV2;

contract Salary {

uint256 hra;

uint256 da;

uint256 bs;

function salcal(uint x) public{

bs=x;

if(bs<=10000){

hra=bs \* 20/100;

da=bs \* 80/100;

}

else if(bs>=10001 && bs<=20000){

hra=bs \* 25/100;

da=bs \* 90/100;

}

else{

hra=bs \* 30/100;

da=bs\* 95/100;

}

}

function fin() public returns(uint256){

uint256 gross=bs+da+hra;

return gross;

}

}

**SET 8**

**2)Write a program in solidity toFind Area and Circumference of a Circle , Rectangle & square.**

**CODE: -**

pragma solidity ^0.5.0;

contract Types {

uint l = 50;

uint w = 25;

uint s = 9;

uint r = 5;

function area\_of\_rectangle() public returns(uint, uint)

{

uint area = l \* w;

uint perimeter = 2 \* (l + w);

return(area,perimeter);

}

function area\_of\_sqaure() public returns(uint, uint)

{

uint area = s \* s;

uint perimeter = 4 \* s;

return(area,perimeter);

}

function area\_of\_circle() public returns(uint,uint)

{

uint area = 3 \* r \* r;

uint circumference = 2 \* 3 \* r;

return(area,circumference);

}

}

**SET 11**

**Dear you have to solve two question 15 marks each**

**1)Write a program in solidity to calculate the Fibonacci Series up to n terms.**

**CODE**

**2)Implementation of SHA-256 with your name**

**CODE**

import hashlib

str = "Enter your name"

# Encoding and sending to SHA256

result = hashlib.sha256(str.encode())

# printing the equivalent hexadecimal value.

print("The hexadecimal equivalent of SHA256 is : ")

print(result.hexdigest())

print ("\r")