**Charotar University of Science & Technology (CHARUSAT)**

**Devang Patel Institute of Advance Technology & Research (DEPSTAR)**

**Blockchain Technology (CE471)**

**Practical Solution**

|  |  |
| --- | --- |
| **Semester:7th** | **Academic Year: 2021-22** |

|  |  |
| --- | --- |
| **Sr No** | **Aim** |
| **1.**  **2.**  **3.** | Install a Bitcoin/Ethereum wallet. Generate and secure your private key. Perform following tasks:  a. Create Account/Import Account  b. Add ether into account  c. Perform transaction to different account  **Solution:**  a. Create Account/Import Account              b. Add ether into account        c. Perform transaction to different account            Write a simple Smart Contract, compile, deploy it using Metamask Ropsten Test Network   * Setting a Greeting —we should be able to set a greeting * Displaying the Greeting —we should be able to retrieve the greeting and display it.   **Solution:**  pragma solidity 0.5.3;  contract TestContract{  string private greeting;  constructor() public{  greeting = "Hello Kashyap Shah!";  }  function setGreeting(string memory newGreeting) public {  greeting = newGreeting;  }  function greet() public view returns (string memory){  return greeting;  }  }        **CONCLUSION:** In this practical, I have learnt about solidity programming , about functions and testing .  Write a smart contract of Election with following tasks:  a. Cast vote against two candidates  b. Vote should be cast only once.  c. No voter can vote any third candidate.  **Solution:**  pragma solidity 0.4.23;  contract Election {  struct Candidate{  uint id;  string name;  uint votecount;  }  mapping(address => bool) public voters;  mapping(uint => Candidate) public candidates;  uint public candidatesCount;  event votedEvent (  uint indexed\_candidateId  );  constructor () public  {  addCandidates("BJP");  addCandidates("Congress");  }  function addCandidates(string \_name) private {  candidatesCount++;  candidates[candidatesCount] = Candidate(candidatesCount, \_name , 0);  }  function vote(uint \_candidateId) public {  require(!voters[msg.sender]);  require(\_candidateId> 0 && \_candidateId<= candidatesCount);  voters[msg.sender] = true;  candidates[\_candidateId].votecount++;  emit votedEvent(\_candidateId);  }  }    **Conclusion:** We learned about Smart Contract of Election. |