.Install and configure Go Ethereum and the Mist browser.

Installing GETH (Go Ethereum)

Step 1: Go to website https://geth.ethereum.org/downloads/ Step 2: From stable releases Geth 1.5.8 (kind = installer) Step 3: once downloaded run it then click next

Step 4: Select Geth and Development tools click next Step 5: Select location to install click next

Step 6: Once Installation is finished Click Close and its done Installing Mist Browser

Step 1: https://github.com/ethereum/mist/releases

Step 2: Under Ethereum Wallet and Mist 0.8.9 - "The Wizard" download mist-installer-0-8- 9.exe

Step 3: For installation click, I agree -> next -> install Run Mist

Step 1: Open the Mist from the start menu

Step 2: It will start downloading Blockchain data once you open it Step 3: Once it finishes downloading it is ready to use

Run Geth

Step 1: Open CMD

Step 2: Type GETH and press enter

Step 3: After it finishes loading press ctrl+c to exit the process. Step 4: Now it's ready to use

.Implement and demonstrate the use of the following in Solidity: Variable, Operators, Loops, Decision Making, Strings, Arrays,

Variable, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs, Mappings, Conversions, Ether Units, Special Variables.

Code:

1] Variable

pragma solidity ^0.8.0; contract SolidityTest {

uint storedData; // State variable constructor() public {

storedData = 10;

}

function getResult() public view returns(uint){ uint a = 1; // local variable

uint b = 2;

uint result = a + b;

return result; //access the local variable

}

}

Output:

2] Operators

Code:

pragma solidity ^0.8.0;

// Creating a contract contract SolidityTest {

// Initializing variables uint16 public a = 20; uint16 public b = 10;

// Initializing a variable

// with sum

uint public sum = a + b;

// Initializing a variable

// with the difference uint public diff = a - b;

// Initializing a variable

// with product

uint public mul = a \* b;

// Initializing a variable

// with quotient

uint public div = a / b;

// Initializing a variable

// with modulus

uint public mod = a % b;

// Initializing a variable

// decrement value uint public dec = --b;

// Initializing a variable

// with increment value uint public inc = ++a;

}

Output:

3] Loops

Code:

pragma solidity ^0.8.0;

contract loop{ uint[4] public arr; uint public count;

function looptest() public{ while(count<arr.length){

arr[count] = count; count++;

}

}

}

Output:

4] Decision Making

Code:

// 'if...else' statement pragma solidity ^0.8.0; contract decisionMaking{

function check(int a) public pure returns(string memory){ string memory value;

if(a>0)

{

value = "Greater than zero";

}

else if(a==0)

{

value = "Equal to zero";

}

else

{

value = "Less than zero";

}

return value;

}

}

Output:

5] Strings and Enum

Code:

pragma solidity ^ 0.8.0;

// Creating a contract contract stringtest {

// Initializing String variable

string public str = "GeeksforGeeks";

// Defining an enumerator

enum my\_enum { geeks\_, \_for, \_geeks }

// Defining a function to return

// values stored in an enumerator function Enum() public pure returns( my\_enum) {

return my\_enum.\_geeks;

}

}

Output:

6] Array

Code:

pragma solidity ^0.8.0; contract Array{

uint[4] public arr =[10,20,30,40];

}

Output: