Assignment No.3

Aim:

Prepare your build system and Building Bitcoin Core.

a. Write Hello World smart contract in a higher programming language (Solidity).

b. Solidity example using arrays and functions

Theory:

- Introduction to smart contract
- Introduction to Solidity programming language

What is Solidity?

- Solidity is a rather simple language deliberately created for a simplistic approach to tackle real world solutions. Gavin Wood initially proposed it in August of 2014. Several developers of the Ethereum chain such as Christian Reitwiessner, Alex Beregszaszi, Liana Husikyan, Yoichi Hirai and many more contributed to creating the language. The Solidity language can be executed on the Ethereum platform, that is a primary Virtual Machine implementing the blockchain network to develop decentralized public ledgers to create smart contract systems.

Solidity Basics

- To get started with the language and learn the basics let's dive into coding. We will begin by understanding the syntax and general data types, along with the variable data types. Solidity supports the generic value types, namely:
- Booleans: Returns value as either true or false. The logical operators returning Boolean data types are as follows:
- ! Logical negation

- && logical conjunction, “and”
- || logical disjunction, “or”
- == equality
- != inequality

Integers: Solidity supports int/unit for both signed and unsigned integers respectively. These storage allocations can be of various sizes. Keywords such as uint8 and uint256 can be used to allocate a storage size of 8 bits to 256 bits respectively. By default, the allocation is 256 bits. That is, uint and int can be used in place of uint256 and int256. The operators compatible with integer data types are:

- Comparisons: <=, <, ==, !=, >=, >. These are used to evaluate to bool.
- Bit operators: &, |, ^ bitwise exclusive ‘or’, ~ bitwise negation, “not”.
- Arithmetic operators: +, -, unary -, unary +, *, /, % remainder, ** exponentiation, << left shift, >> right shift.

The EVM returns a Runtime Exception when the modulus operator is applied to the zero of a “divide by zero” operation.

Address: An address can hold a 20 byte value that is equivalent to the size of an Ethereum address. These address types are backed up with members that serve as the contract base.

String Literals: String literals can be represented using either single or double quotes (for example, "foo" or 'bar'). Unlike in the C language, string literals in Solidity do imply trailing value zeroes. For instance, “bar” will represent a three byte element instead of four. Similarly, in the case of integer literals, the literals are convertible inherently using the corresponding fit, that is, byte or string.

Modifier: In a smart contract, modifiers are used to ensure the coherence of the conditions defined before executing the code.

Solidity provides basic arrays, enums, operators, and hash values to create a data structure known as **"mappings**." These mappings are used to return values associated with a given storage location. An Array is a contiguous memory allocation of a size defined by the programmer where if the size is initialized as K, and the type of element is instantiated as T, the array can be written as T[k].

Arrays can also be dynamically instantiated using the notation uint[][6]. Here the notation initializes a dynamic array with six contiguous memory allocations. Similarly, a two dimensional array can be initialized as arr[2][4], where the two indices point towards the dimensions of the matrix.

We will begin our programming venture with a simple structure of a contract. Consider the following code:

```
pragma solidity^0.4.0;
contract StorageBasic {
  uint storedValue;
  function set(uint var) {
     storedValue= var;
  }
  function get() constant returns (uint) {
   return storedValue;
  }
}
```

- The word "**Pragma**" refers to the instructions given to a compiler to sequentially execute the source code.
- Solidity is statically typed language. Therefore, each variable type irrespective of their scope can be instantiated at compile time. These elementary types can be further

combined to create complex data types. These complex data types then synchronize with each other according to their respective preferences.

- What are different data types and variables in solidity.
- Syntax of a program in solidity.
- How to declare arrays, functions and structures in solidity

Implementation:

- Installation of Remix and introduction to all sections in remix IDE
- Write a Smart contract to display hello world:
  - https://solidity-by-example.org/hello-world/
- Connect and deploy smart contract using metamask.
- Write and execute a smart contract which uses array, function and structure in it.

Conclusion:

FAQs:

1. What are some important features of Solidity?
2. What types of applications can be developed using Solidity?
3. What are the main differences between Solidity and other programming languages like Python, Java, or C++?
4. What is EVM bytecode?
5. What tools can be used for testing Solidity codes?
6. What is a smart contract's ABI?
7. Is Solidity frontend or backend?