Questions:

A. Name two value types that are known in Solidity Programming?

- Boolean/ Integer

B. Name two reference types that are known in Solidity Programming?

- fixed-sized arrays/ dynamic-sized

C. What are globally available variables in this scope?

- pragma solidity^0.4.0;

D. Name two types of `block` variables.

- block.number (uint): current block number
- block.timestamp (uint): current block timestamp as seconds since unix epoch

E. Name one form of visibility in Solidity that is NOT AVAILABLE in C++.

- A function's visibility can be set to external, public, internal or private, while state variables only have three possible visibility modifiers; public, internal or private. The keyword external is not applicable to state variables.

F. Provide me one example of using the SafeMath with Inheritance.

```
// Solidity program to
// demonstrate
// Single Inheritance
pragma solidity >=0.4.22 <0.6.0;

// Defining contract
contract parent{

    // Declaring internal
    // state variable
    uint internal sum;

    // Defining external function
    // to set value of internal
    // state variable sum
    function setValue() external {
        uint a = 10;
        uint b = 20;
        sum = a + b;
    }
}

// Defining child contract
contract child is parent{

    // Defining external function
    // to return value of
    // internal state variable sum
    function getValue(
    ) external view returns(uint) {
        return sum;
    }
}

// Defining calling contract
contract caller {

    // Creating child contract object
    child cc = new child();

    // Defining function to call
    // setValue and getValue functions
    function testInheritance(
    ) public returns (uint) {
        cc.setValue();
        return cc.getValue();
    }
}
```

G. Provide me one example of using the SafeMath with the builtIn Library.

```
pragma solidity ^0.7.0;

import
"https://github.com/OpenZeppelin/openzeppelin-solidity/contracts/m
ath/SafeMath.sol";

contract MyContract {
  using SafeMath for uint;

  uint public balance;

  function deposit(uint amount) public {
    balance = balance.add(amount);
  }
  function withdraw(uint amount) public {
    require(amount <= balance, "Insufficient balance");
    balance = balance.sub(amount);
  }
}
```

H. Take a look at this code contract information, because I will be asking you about it on my return, January 9th.

https://github.com/0xProject/0x-monorepo/tree/development/packages/contracts