October Contest

Competition link: TODO

Warning:

This contest assumes that you:

+ Have taken the CS course here at Harrison
+ Are currently taking the CS course at Harrison
+ Plan to be/are a part of the Competitive Programming Club.

If not, the material may be more confusing than normal.

Contents

What is blockchain and cryptocurrency - Slide 4

Solidity Tutorial TODO

Examples TODO

What is the blockchain?

What is a business transaction?

+ **<u>One party receives goods from another party in exchange for other goods.</u>**

I.e. You buy a coffee

You are one party

Starbucks or wherever you get your coffee is another party

You exchange money (cash/debit/credit …) in exchange for the coffee.

+ This is a business transaction.

Third parties in business transactions

+ The simplest business transaction is a direct exchange between two parties.

I.e. Alice trades Bob a pencil for a pen.

+ However, most transactions today involve a **third party** s/a a bank, a company, an app that does things that affect a transaction.

I.e. Alice buys coffee with her credit card. The credit card company is a **third party**.

Risks with third parties

+ This can be dangerous!

I.e. What happens if your credit card company decides to stop supplying money?

+ In the real world, there are rules and laws that protect us against these situations, however, *they are not foolproof*.
+ There is always a risk when third parties appear in business transactions. Especially if they can do things that we cannot see.

I.e. Your credit card company suddenly starts collecting a hidden tax on every purchase.

What is transparency?

+ ***Transparency*** in finance is when the actions taken by a party are visible to everyone.

I.e. Chase bank imposes a new tax on all purchases but does not make an announcement. This tax would be a ***non-transparent act***, as the customers of Chase were not notified of this change, and some may not even notice the tax.

On the other hand, if Chase bank made an announcement to the public about their new tax, their tax would be a ***transparent act***, as customers are aware of this action.

<u>Third parties are a risk if their actions are not transparent!</u>

What is centralized finance?

+ The term for third parties in business transaction is ***centralized finance***.
+ The idea is that everything, is *centered* around a third party that provides some service

I.e. Many people depend on the Chase company for their credit and debit cards, so the Chase company is a ***centralized finance*** .

What is decentralized finance?

+ On the other hand, ***decentralized finance*** is when there are no third parties involved in a transaction, or, anything that a third party does is **transparent**
+ Also known as ***DEFI***

Examples of DEFI

Bitcoin

Ethereum

Dogecoin

Most NFTs (Non-fungible Tokens, think of the weird monkey pictures that have been showing up lately)

Most DeFIs run off of a blockchain

What is a blockchain?

+ Official whitepaper: https://bitcoin.org/bitcoin.pdf (don’t have to read, but recommended anyway)
+ ***A blockchain is a publicly visible, append-only ledger with a multiple-party consensus system.***
+ Try to read through those words one-by-one and see if they makes sense to you.
+ Simple explanation follows.

A blockchain is a publicly visible …

+ Everything on the blockchain is visible to everyone
+ This includes all changes and transactions

I.e. Let's say that I buy an NFT on a blockchain today. Anyone can see that I bought the NFT, from today till the blockchain shuts down.

+ Changes and transactions that are made on the blockchain are known as ***on-chain***

… append-only …

+ Append is a fancy word for “add”
+ You can only add to the blockchain, removing things from the blockchain is impossible.

I.e. My purchase of the NFT from the previous slide is on-chain permanently, even if I sell my NFT to someone else.

+ Think of it like your search history, but without the delete history option.

… ledger …

- Ledgers are books used to keep track of transactions.

I.e. Bank ledgers

- This means that the blockchain is basically a virtual ledger that keeps track of all of the transactions.

… with a multiple-party consensus system (I)

- This is the part that I expect no one to understand (if you did, you are either a genius or have experience)
- As mentioned earlier, the blockchain is publicly visible, meaning that everyone can see and make changes to the blockchain.
- What if someone tries to make a bad change?

I.e. A new 99% tax on all purchases, delete blockchain, …

… with a multiple-party consensus system (II)

- The blockchain resolves this using a sort of voting system.
- People known as ***miners*** vote on whether or not a change should be made.

I.e. Most miners would probably be against the 99% tax, thus they would vote to repeal the transaction **BEFORE IT IS ADDED TO THE BLOCKCHAIN**

- Miners are usually paid a fee for their services

I.e. bitcoin miners, dogecoin miners

- There are specific rules on miners and how they operate, but that is not too important for us.

What is a blockchain?

+ Putting all the previous slides together:
+ A blockchain is a virtual ledger, where anyone can make changes or transactions.
+ These changes and transactions are approved by people known as miners, who vote on whether or not a change should or should not be added to the blockchain.

Examples of blockchains

Ethereum

Solana

Polygon

The advantage of the blockchain

+ Remember one of the essential components of the blockchain: **It is publicly visible**
+ Any changes by third parties must be done through transactions, which are **publicly visible**
+ Therefore, blockchains are **transparent**, which is a good thing.
+ Formally, blockchains are decentralized finances (DeFI)

Things on the blockchain

+ Transactions can be for anything: paying for something, buying something, deploying something…
+ Betting games, Stock markets, Staking, Marketplaces, etc are all existing deployments on the chain.
+ This means that anyone who accesses the blockchain can interact with these games, markets, whatever.

But how do such things get deployed onto the blockchain?

Contracts

+ Any bigger systems such as betting games, etc, are written as ***contracts***
+ Simply put, **contracts** are collections of code that support various **functions** that do different things (like Python).

I.e. In a banking contract, one function could be to **deposit** money, and another function could be to **withdraw** money.

+ Contracts are usually written in **Solidity**, a special language

What is Solidity? (Solidity Tutorial Part I)

https://www.tutorialspoint.com/solidity/index.htm

You will have to submit code in Solidity

Getting a working environment:

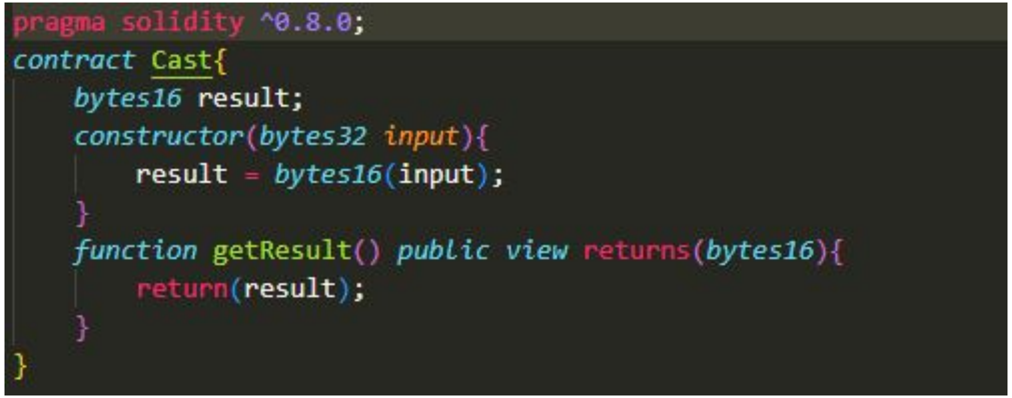
- Follow the steps in the link above
- I personally use vscode

Basic Overview:

Solidity

+ Solidity is a language. Similar to Python, Java, C, C++ …
+ It is most similar to Java

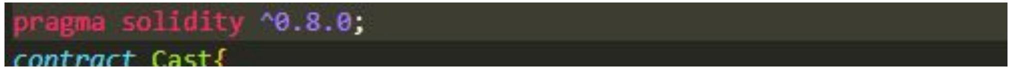
I.e. Sample Code (can you tell what it does?):



```
pragma solidity ^0.8.0;
contract Cast{
    bytes16 result;
    constructor(bytes32 input){
        result = bytes16(input);
    }
    function getResult() public view returns(bytes16){
        return(result);
    }
}
```

Initializing Solidity

+ The first line in any Solidity file is usually pragma solidity XXX
+ This specifies what version of Solidity is being used. (Kinda like Iphone models)
+ The most recent version is around 0.8.15.



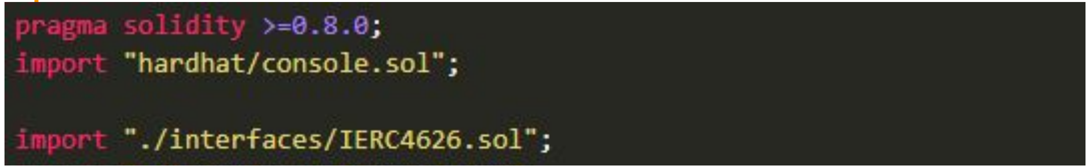
```
pragma solidity ^0.8.0;
contract Cast{
```

+ The '^' character means that the file is using a version of Solidity at or above the currency version.

I.e. ^0.8.0 means that it is any version (0.8.0, 0.8.1, 0.8.2, …)

Import files

+ Next, files to import are added.
+ Since there are many things on the blockchain, sometimes it is better to use something that someone else made instead of making it yourself.
+ Therefore, you can import files by writing import “FILEPATH”
+



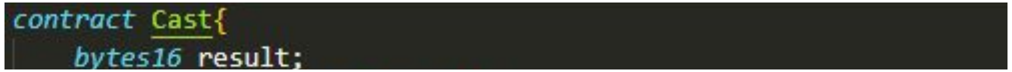
```
pragma solidity >=0.8.0;
import "hardhat/console.sol";

import "./interfaces/IERC4626.sol";
```

+ One important thing: **“hardhat/console.sol”** is the `print` statement in Solidity (meaning that it makes things show up on the screen).

Define Contract

+ Next, the main file is defined.
+ In Solidity, this is known as the contract
+ To do this, the line contract ContractName is written

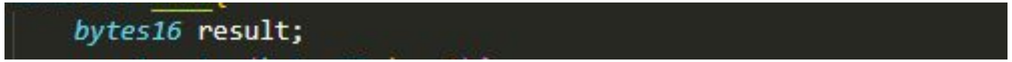
I.e. 

```
contract Cast{
    bytes16 result;
```

Defines a contract named “Cast”

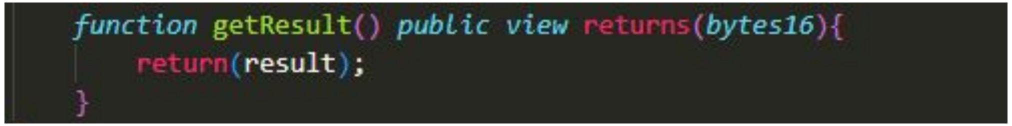
The essential parts of a contract

- There will be more expanded on this later, but for now, the most important thing is that: Contracts are made up of two things: **functions** and **variables**.
- **Variables** act like storage

  I.e. result is a variable: 

```
    bytes16 result;
```

- **Functions** act like … functions (They do things that can change the variables)

  I.e. getResult() is a function: 

```
    function getResult() public view returns(bytes16){
        return(result);
    }
```

Variables

Naming Variables

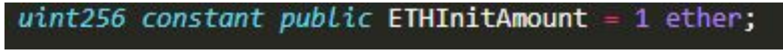
- (Simple) Variables have a few main parts:
  - **Type (1)**
  - **Name (2)**
  - **Value (3)**

I.e. int i = 0;

“int” is the type (it stands for integer, which is essentially a number)

“i” is the name

“= 0” sets the value of the variable to 0.



```
uint256 constant public ETHInitAmount = 1 ether;
```

Using and Setting Variables

- Like in math, anytime that you write “i”, that corresponds to the value 0.
- There are a few common types in all languages:
    - Integers (uint)
    - Strings (String)
    - Boolean (boolean)

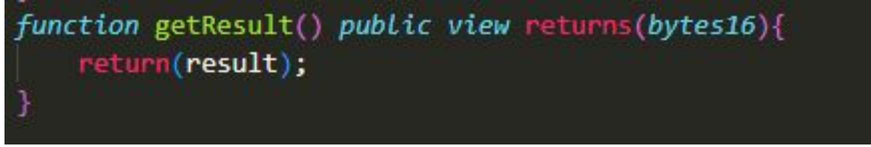
I.e.

```
uint myInteger = 5;     //define a variable myInteger = 5

String myString = “hello”;   //define a variable myString = “hello”

boolean myBool = true;    //define a variable myBool = true;
```

Functions



```
function getResult() public view returns(bytes16){
    return(result);
}
```

Functions

- Solidity Functions are like functions in math (f(x) = y)
- Takes “x” and returns “y”
- Function formula:
- “function” name(inputs) visibility returns(output type){

    //internal changes

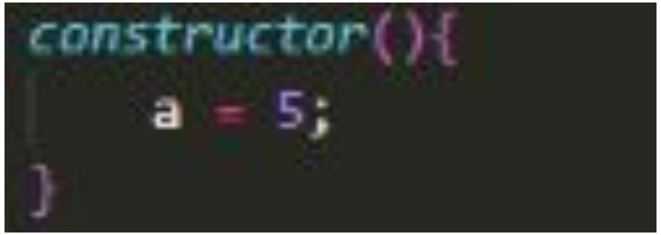
    return(output)

  }

- The function in the picture is named “getResult”, with no input, and returns the variable “result”, which is a byte16

Define constructor()

+ Constructors are functions that are called once and only once when the contract is deployed.



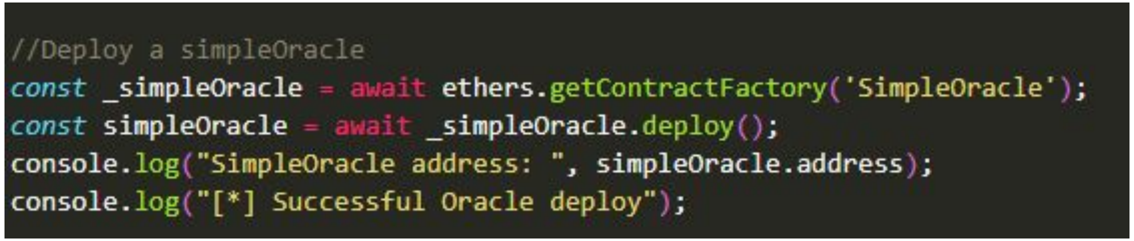
```
constructor(){
    a = 5;
}
```

Tip 1 of auditing:

- Functions are the most important thing in auditing
- Make sure you know exactly what each function is doing
- 90% of auditing issues come from functions not doing what they are intended to do.

Solidity files deployment (Out of Scope)

+ When you finish solidity files, they need to be deployed.
+ Once deployed, anyone can access functions and variables in a contract.



```
//Deploy a simpleOracle
const _simpleOracle = await ethers.getContractFactory('SimpleOracle');
const simpleOracle = await _simpleOracle.deploy();
console.log("SimpleOracle address: ", simpleOracle.address);
console.log("[*] Successful Oracle deploy");
```

Miscellaneous information: console.log (I)

+ Solidity does not support print, so usually, developers import “hardhat/console.log”
+ Then print using “console.log(“whatever”)
+ Similar to java printf

I.e. console.log(“apples”)   //prints apples

+ Special characters:

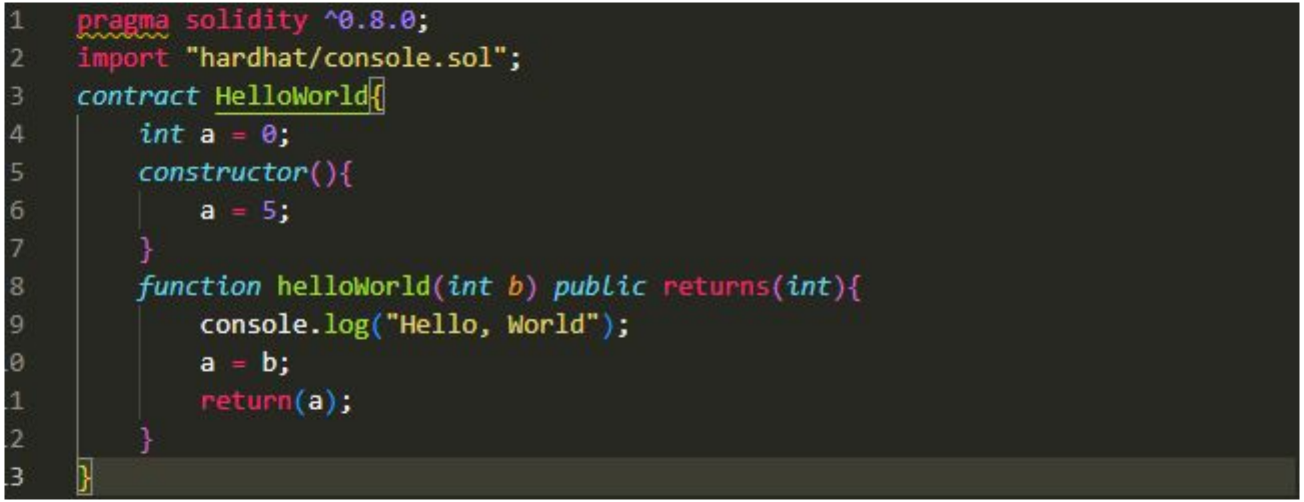
I.e. console.log(“Number %d”, 5) //prints Number 5

%d replaces itself with the number after the comma.

http://web.mit.edu/10.001/Web/Course_Notes/c_Notes/tips_printf.html

Examples of Solidity

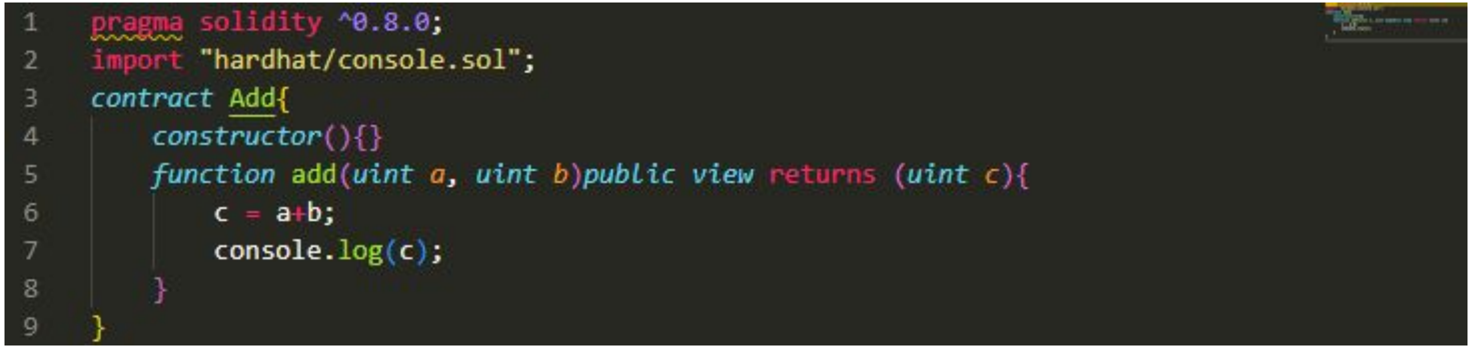
Helloworld.sol



```
pragma solidity ^0.8.0;
import "hardhat/console.sol";
contract HelloWorld{
    int a = 0;
    constructor(){
        a = 5;
    }
    function helloWorld(int b) public returns(int){
        console.log("Hello, World");
        a = b;
        return(a);
    }
}
```

After contract HelloWorld is deployed, when function “helloWorld” is called, “Hello, World” is printed and a is set to b and returned.

Add.sol



```
pragma solidity ^0.8.0;
import "hardhat/console.sol";
contract Add{
    constructor(){}
    function add(uint a, uint b)public view returns (uint c){
        c = a+b;
        console.log(c);
    }
}
```

After contract Add is deployed, when function “add” is called with uints a and b as inputs, the function defines uint c as a+b, and prints and returns c.

Rubric:

Short Answer: 20 pts (all or nothing)

Screenshot of Working Environment: 10pts (all or nothing)

Code1: 35pts

- Attempted: 20pts (provide screenshot)
- 5pts functionA
- 5pts functionB
- 5pts functionC

Code2: 35pts

- Attempted: 20pts (provide screenshot)
- 5pts gotResultA
- 5pts gotResultB
- 5pts gotResultC