COSC 5010 - Blockchain Design and Programming Lecture Note 2

Libao Jin (ljin1@uwyo.edu)

October 24, 2018

1 Lecture 01 - Aug 29, 2018

1.1 Overview

1.1.1 Lecture 01 - Aug 29 - Wednesday - Class Overview

1. A little bit of background on the instructor

    1. pschlump@uwyo.edu
    2. https://github.com/Univ-Wyo-Education/Blockchain-4010-Fall-2018
    3. (for emergencies between 7:00AM and 9:00PM) 720-209-7888 (cell)
    4. This is a "practical" class.
    5. Class Goal - have every student in this class be able to work effectively in the Blockchain/Ethereum world. There are students in this class that are from other departments. The class has been structured to take this into account.
    6. Where the students are and the opportunity they have today.
    7. Realizing Your Dreams.

2. Class policy - UW requires that I talk about cheating. stackoverflow.com and Google are your fiends in this class - give credit where credit is due. warning: stackoverflow.com on ethereum/solidity is badly out of date becuse the technology is changing really fast. If you copy from the web - I expect a comment and a link (URL) to the source of where you got your copy. I expect an explanation of what and why you are grabbing someting from the web. I expect an analysis of what license the content is under, MIT, GPL2, GPL3, CC, CC-BY etc.

3. Late Policy and (3) Late Coupons I have a grader - most of the grading will be automated. You get 3 late coupons - I will send them to you in email. You can use them for handing in homework 1 week after the due date. Late coupons will be tracked on a semi-private Ethereum Blockchian with a Smart Contract.

    Application: http://www.2c-why.com/UW-dApp/

4. 70% from homework, 40% from tests. Midterm and Final. There will be 2 projects for the class. You will have to give a live, in person, demo of both of your projects and you **must** get them to work. They are required to pass the class.

5. Class Points Total

    Points available:

    []@rl@

Points	Class Item
1400	Homework total800 2 tests

    To get a letter grade in the class:

    []@ll@

Points	Semester Grade
1900 or above	A+
1800 … 1899	A
1600 … 1799	B
1400 … 1599	C
1200 … 1399	D

You **must** demonstrate working projects to the instructor to pass the class (no matter how many points you get). The 2 projects in the class will be directly from the homework. Project one is Homework Assignments 02 to 06. Project 2 is Homework Assignments 08 to 11. Homework will be 100 or 200 points each. Midterm will be 400 points. Final will be 400 points and cumulative. You can expect programming and writen assignments in this class.

For anybody that just wants to take on a hard project for extra credit see the instructor. It is hard. So think a letter grade for completion of an extra credit project. Code for extra credit projects will be open source under a MIT license. Also note that there are 2200 points available on a letter grading scale of 2000 points. You have a built-in 200 point extra credit in the homework and tests.

6. textbook: An Introduction to Programming in Go, pdf for free There are no good books on Ethereum/-Soliddity. Solidity has moved from version 4.12 to 4.27 this year. All of the books are out of date. So…. I will include links in assignments that you are expected to read.

7. Class Overview

    1. What is Blockchain - what is Bitcoin / Ethereum / Other token systems
    2. The worlds worst, most expensive database
    3. What is the “hype” - what is real.
    4. Economics - Coin, ICO, Stocks, Bonds, Tokens, Utility Tokens, A Security
    5. Legal Ramifications. ICOs 506(d), Subpart (s)
    6. Programming - 1/2 in go, 1/2 in Solidity (Etherem) and Web front end (JavaScript/HTML/CSS).
    7. Some Homework
    8. Write a Paper - How will blockchain effect the economy.
    9. 2 tests (Midterm and Final)
    10. Why Go
    11. Proof of work
    12. Proof of stake
    13. Enough Go to make it through this class (and be able to convincingly tell an employer that you have programmed in Go)
    14. Why Ethereum? Solidity?
    15. dApp - what is that? What is web3?
    16. A detailed understanding of the security model behind Blockchain
    17. Some advanced stuff on security - distributed computation and public/private keys, distributed key generation.
    18. What is a “tangle”
    19. Why is blockchain so slow?
    20. How to explain “blockchain” to people - the 30 second elevator pitch.

8. De-Hype Blockchain

    1. Successes
        - Show URLs
        - Forcast
    2. Failures
        - Over 1/2 of ICOs are proably fraud

2 Lecture 02

2.1 Question: What is a hash?

1. (math) a mapping form a range to a domain.

2. Text to a number.

```
abc                275f20377d6574b67399702947cb56849d2e02f7112c1d021603346c345b37f8
abd                3212601953780d3a8de118531b87bf6183edb8c81baf6982fdca260033a5f29b
war-and-peace.txt 67c570b0e09d70225d739aec9a7ea91631df1ea06ba44f0c9d9fe99e45f41756
```

3. Different kinds of hash, MD5, SHA1, SHA256, SHA3, SHA512, Keccak256.

2.2 attestation - Definition of attestation - From Websters Dictionary.

https://www.merriam-webster.com/dictionary/attestation Faire Use - this is a quote.

1. an act or instance of attesting something: such as
    1. : a proving of the existence of something through evidence … a complete and formal attestation of your innocence. —Edward Bulwer-Lytton
    2. : an official verification of something as true or authentic the notary's attestation of the will
2. the proof or evidence by which something (such as the usage of a word) is attested the earliest attestation of the term in print

2.3 Economics of Blockchain

In 2009 - bitcoin invented. In 2013 - smart contracts - Ethereum.
Ability to create trust between non-trusting parties. Ability to create economic systems.
Merick - shipping 40% decrease in shipping times. World wide $380 Billion in trade. 90% of all the goods in the world are moved by ship. 38 days average shipping time. A 40% reduction is dropping that to 23 days. Difference is 15 days. 40% of 380 billion is 152 billion in capital that is not tied up - at an average capital cost of 10% = 15.2 billion - over the 23 days. 15/365 * 15.2 billion. – About 0.62 billion.
Estonia - All titles and property on the chain. In US 5.51 Million Houses. Average title search $3821. Title search in Estonia, $23.33 each. Title insurance $1408. My calculation $28 billion a year.
Marshal Islands - off of the dollar and onto a blockchain. The estimate is that the government will save around $5M a year or about $90 per resident a year. For entire US 327 million - that is $29 billion dollars.
Australian University is issuing certificates based on blockchain. (Especial Interest in this because this is our project)
Perspective on 28 or 29 billion - free college tuition for all students in the United States is estimated to cost 75 billion.

3 Go - Intro

Assignment 1 - Due Sep 12

1. Hello World - walk through
2. Echo - walk through
3. Marshal and Unmarshal of data - walk through

4 More On Go

"It is not the strongest of the species that survies, nor the most intelligent. It is the one that is the most adaptable to change." Charles Darwin

4.1 1st. the news

1. AU - Research - linear scaling (similar to blockchain) 8000 tx per sec, don't specify latency.
2. Java 4th garbage collector - tried it. Provision at 20% before, now at 25% ok, at 30% boom.
3. Did everybody in the class get a "key".
4. California bill passed that defines things like "digital signature".

4.2 2nd. Purpose of a business

To make a profit for the owners of the business.
What is "Fiduciary Responsibility". It means that you have been placed / are in a position of legal responsibility for managing somebody else's money.

4.3 Go

1. Functions

Parts of a function

```
// Name will do a b c d.
func Name ( i1 Type1, i2 Type2 ) ( o1 Type1, o2 Type2 ) {
    // Comment in body
}
```

Observations 1. Name starts with a capital, therefore exported from package. 2. You can have more than 1 return value. 3. "error" is usually the last return value. 4. Comments on functions need to start with the name of the function.

2. Maps

Go has dictionary/maps

```
var m1 map[string]int
m1 = make(map[string]int)
m1["abc"] = 12
k := m1["abc"]
k2 := m1["xyz"]
k3, ok_t := m1["abc"]
k4, ok_f := m1["xyz"]
```

Observations 1. memory is not allocated to a map when it is declared. 2. You can just use `make` and `:=` to declare a map. 3. You can test to see if you have an un-allocated map by comparing to `nil`. 4. You can find out if a value is in a map.

3. Slices (Arrays)

An Array

```
var a1 [4]int
```

A slice

```
var s1 []int
```

What is a slice?

4. io & cli

Output:

```
fmt.Printf ( "Format %s\n", "string" )
```

CLI

```
fmt.Printf ( "%s\n", os.Args[] )
```

5 Blockchain and Mining

1. News

    1. Hack-a-Thon has picked up an additional $20,000 challenge. (You can still sign up at the door today)
    2. Goldman Sacks - is not doing bitcoin trading, Bitcoin down 10 %, Eth under $200 a token.
    3. SEC's internal review of ICOs suggests that about 1/2 of them are fraudulent.
    4. Next Week - assignment 1 is due.
    5. UW has finally gotten my account set up to use the "wyocourses" stuff - so PDF's and other things will be on the Learning Management System.
    6. 10th (Monday) we will be sending out links for the "late coupons".

2. Definitions / Economics

"arbitrage" ::= the simultaneous buying and selling of securities, currency, or commodities in different markets or in derivative forms in order to take advantage of differing prices for the same asset.

2. How Soros broke the Bank of England.

Price Stable Cryptocurrences (or pegged cryptocurrencies)

5.1 Reference

Soros Broke the Bank of England

3. What is Mining and How is it implemented.

What is proof-of-work? What is proof-of-stake? What is a public blockchian / private blockchain?
Diagram of blocks
Diagram of mining
What is the mining process

4. More on Go

Maps do not synchronize automatically. So… Synchronization Primitives:
[]
package main
import ("fmt" "sync" "time")
// SafeCounter is safe to use concurrently. type SafeCounter struct { v map[string]int mux sync.Mutex }
// Inc increments the counter for the given key. func (c *SafeCounter) Inc(key string) { c.mux.Lock() // Lock so only one goroutine at a time can access the map c.v. c.v[key]++ c.mux.Unlock() }
// Value returns the current value of the counter for the given key. func (c *SafeCounter) Value(key string) int { c.mux.Lock() // Lock so only one goroutine at a time can access the map c.v. defer c.mux.Unlock() return c.v[key] }
func main() { c := SafeCounter{v: make(map[string]int)} for i := 0; i < 1000; i++ { go c.Inc("somekey") }
time.Sleep(time.Second) fmt.Println(c.Value("somekey")) }

5.1.1 A Go Core/Panic

First the Code
[] package main
import "fmt"
var mm map[string]int
func main() { fmt.Println("vim-go") mm["bob"] = 3 }
Then the bad output.

```
panic: assignment to entry in nil map

goroutine 1 [running]:
panic(0x10a5540, 0x10d03a0)
    /usr/local/Cellar/go/1.10.3/libexec/src/runtime/panic.go:551 +0x3c1 fp=0xc42005fec8 sp=0xc42005fe28 pc=0x1025
runtime.mapassign_faststr(0x10a4e80, 0x0, 0x10be68a, 0x3, 0x0)
    /usr/local/Cellar/go/1.10.3/libexec/src/runtime/hashmap_fast.go:696 +0x407 fp=0xc42005ff38 sp=0xc42005fec8 pc
main.main()
    /Users/corwin/go/src/github.com/Univ-Wyo-Education/Blockchain-4010-Fall-2018/Lectures/Lect-04/samp.go:9 +0x92
runtime.main()
    /usr/local/Cellar/go/1.10.3/libexec/src/runtime/proc.go:198 +0x212 fp=0xc42005ffe0 sp=0xc42005ff88 pc=0x10278
runtime.goexit()
    /usr/local/Cellar/go/1.10.3/libexec/src/runtime/asm_amd64.s:2361 +0x1 fp=0xc42005ffe8 sp=0xc42005ffe0 pc=0x10

goroutine 2 [force gc (idle)]:
runtime.gopark(0x10c5580, 0x11397a0, 0x10bfafe, 0xf, 0x10c5414, 0x1)
    /usr/local/Cellar/go/1.10.3/libexec/src/runtime/proc.go:291 +0x11a fp=0xc42004a768 sp=0xc42004a748 pc=0x1027d
runtime.goparkunlock(0x11397a0, 0x10bfafe, 0xf, 0x14, 0x1)
    /usr/local/Cellar/go/1.10.3/libexec/src/runtime/proc.go:297 +0x5e fp=0xc42004a7a8 sp=0xc42004a768 pc=0x1027db
runtime.forcegchelper()
    /usr/local/Cellar/go/1.10.3/libexec/src/runtime/proc.go:248 +0xcc fp=0xc42004a7e0 sp=0xc42004a7a8 pc=0x1027b4
runtime.goexit()
    /usr/local/Cellar/go/1.10.3/libexec/src/runtime/asm_amd64.s:2361 +0x1 fp=0xc42004a7e8 sp=0xc42004a7e0 pc=0x10
created by runtime.init.4
    /usr/local/Cellar/go/1.10.3/libexec/src/runtime/proc.go:237 +0x35
```

"assignment to entry in nil map"
..."/Lect-04/samp.go:9"...

6 Lecture 5 - More on Mining, Go Interfaces and Go Weaknesses

6.1 News

1. Hack-A-Thon
2. Hack-A-Thon
3. China is accepting digital digests (hash) as evidence in a court of law.
4. Australia is putting drivers licenses on the blockchain

6.2 Detailed walk through of mining.

Walk Through

1. Use an infinite loop to:
   1. Serialize the data from the block for hashing, Call `block.SerializeForSeal` to do this.
   2. Calculate the hash of the data, Call `hash.HashOf` to do this. This is the slow part. What would happen if replaced the software with a hash calculator on a graphics card where you could run 4096 hahes at once? What would happen if we replaced the graphics card with an ASIC - so you had dedicated hardware to do the hash and you could run 4 billion hashes a second?
   3. Convert the hash (it is []byte) to a hex string. Use the `hex.EncodeToString` standard go library function.
   4. `fmt.Printf("((Mining)) Hash for Block [%s] nonce [%8d]\r", theHashAsAString, bk.Nonce)`
   5. See if the first 4 characters of the hash are 0's. - if so we have met the work criteria. In go this is `if theHashAsAString[0:4] == "0000" {`. This is create a slice, 4 long from character 0 with length of 4, then compare that to the string `"0000"`.

```
   - Set the block's "Seal" to the hash
   - `fmt.Printf("((Mining)) Hash for Block [%s] nonce [%8d]\n", theHashAsAString, bk.Nonce)`
   - return
  5. Increment the Nonce in the block, and...
  6. Back to the top of the loop for another try at finding a seal for this block.
```

6.3 Go Interfaces

Two uses for interfaces (Actually more than 2 but 2 primary uses).

1. Variable parameter list functions.
2. Interfaces to sets of functions.

6.4 Variable parameter list functions.

[] func vexample(a int, b ...interface{}) { for pos, bVal := range b { switch v := bVal.(type) { case int: fmt.Printf("It's an int, case []int: fmt.Printf("It's a slice of int\n") default: fmt.Printf("It's a something else\n") } } }

6.5 Interfaces to sets of functions.

```
type InterfaceSpecType interface {
    DoFirstThing(p1 int, p2 int) error
    DoSomethingElse() error
}

type ImplementationType struct {
    AA int
    BB int
}

var _ InterfaceSpecType = (*ImplementationType)(nil)

func NewImplementationType() InterfaceSpecType {
    return &ImplementationType{
        AA: 1,
        BB: 2,
    }
}

func (xy *ImplementationType) DoFirstThing(p1 int, p2 int) error {
    // ... do something ...
    return nil
}

func (xy *ImplementationType) DoSomethingElse() error {
    // ... do something ...
    return nil
}

func Demo() {
    var dd InterfaceSpecType
    dd = NewImplementationType()
    _ = dd.DoSomethingElse()
}
```

6.6 Go Weaknesses

What are the limitations of using Go

1. No objects - Use interfaces instead. No inheritance.
2. No generics - Use templates and code instead.
3. No error handling - Just return errors.

Go 2.0 is coming in 1.5 years. Go's design team commitment is 100% backward compatibility - it will be able to correctly compile go 1.0 code without change to the language.

7 Merkle Trees

I thought that I would do Merkle trees in this lecture. But we are going to do an overview of transactions instead. Merkle trees are used to validate that a set of transactions has not changed - and that the data is real. Let's start with the reason for using the Merkle trees first. Then on Monday we will get to the Merkle trees.
So…
This is the stuff that I was going to start on Monday - I have just swapped the order a little bit.
Merkle trees is assignment 3, due Sep 26 - so swapping lectures will not have any adverse effect.
Lat time we talked about blocks and a blockchain. Lot's of Go stuff - interfaces… Stuff like that.
Today Let's get into the nitty-gritty of Blocks and Transactions. This is what bitcoin is all about. This is the "distributed ledger".

1. Diagram of Blocks
2. Diagram of Transactions
3. Diagram of a "send" of funds

8 Merkle Trees

Finally - In detail. Merkle Implementation. Code Walk through.
Economics - Bitcoin/Crypto-Currencies are down $640 billion. What this means.
Is this the end of Blockchain.
History Lesion - Seat of the Gate / Eye of the Needle.
SQL crash.
Dot-com crash.
Normal technology cycles.

8.1 TODO

1. Send out security note. It is in a file at the top of the course.
2. A "minimal" git guide. Write it - send it out.

9 Merkle Trees

More on Transactions. Handout and walk through.

9.1 News

1. More trade tariffs.
2. Harvard's "Assessment of Technological Advancement"
    1. Kinetic Safety Bumpers, 1972 to 1992.
    2. Stanley - 21 years later

3. Human Genome, 2003 - First human trials on Cystic Fibrosis.

3. EU and Crypto Regulation
4. Digital Signature Bill (Wyoming)
5. World Economic Forum - Blockchain to play a critical role in environmental problems.

10 Transactions

Handout…
Walk through of the index and how it is used.

10.1 TODO

1. A "minimal" git guide. Write it - send it out.

11 Lecture Changed

Old: Public/Private keys - I will update the syllabus with a modified schedule. Public/Private keys will be Oct 10.
Will be gone Oct 12 and Oct 19th.
We will do "transactions" again next Monday - 1. New handouts 2. Beta test the lecture before hand 3. Get some outside review of the lecture (Apply the power of may eyes on the subject)

11.1 News

1. Washington state has in the works a digital-notary bill based on PGP signatures.
2. France is setting up to allow raising of funds via ICOs
3. The EU is researching the potential of drafting regulations for EU wide crowd funding that includes ICOs
4. "electronum" over 1 million downloads on 1st day - currently running over 1000 transactions an hour from searching a number of supported systems logs.

11.2 Stocks, ICOs, Bonds, etc...

Terms with explanation:

1. Cash - this is what you buy your pizza with.
2. Stock - partial ownership in a company.
3. what is a P&L - financial statement of income and expenses that shows if the company is making or loosing money.
4. Long on Cash, Long in a Stock - this means that you have lots of cash, or that you own some of a stock.
5. "Take a position in"… - to purchase stock in.
6. Real Estate - Investing
7. What is Inflation?
8. REIT (Real Estate Investment Trust) - legally required to pay out 80% of profits to investors.
9. Price to Earnings Radio - a P/E ratio is the cost of share of the company divided by Earnings of that particular share of a company.
10. What is a Derivative
11. High speed trading
12. Dividends - Payment of profit to investors as "income" or a reward for owning the stock.
13. What is the "Yield"
14. What is a basis point (BPS), Example 50 BSP = 0.5%
15. Stock Buy Back

16. Index Fund
17. Insider Trading
18. Bonds
19. ICOs (Initial Coin Offering)
20. “Industrial Staker Sale”
21. Proof-of-Work, Proof-of-Stake
22. “Consumer Token Sale”
23. Junk Bonds - non investment grade bond. Bonds are rated for default risk.
24. Mutual Fund
25. Asset Allocation
26. Expense Ratio for a Mutual Fund
27. Prospectus
28. Pro-Forma
29. KYI - Know Your Investor (See SEC 506(d))
30. Certified Investor
31. Going public - Make a public offering. Cost etc.
32. Money Laundering
33. Wyoming Utility token LLC (HB-70)
34. Cost of “going public”

12 Redo of Transactions and start on Full Stack

Pass 2 at explaining how transactions work in detail.
Then start on how HTTP and network work.

12.1 News

1. Possible proof to the Reinman hypothesis. If proven has huge implications for RSA and EC based security.
2. England looking at banning crypto currencies. (I think that they already ban torrenting of movies too)
3. Blockchain coming to a CPU near you.
4. Tokenized Wine - as a collectable.

12.2 Tranactions

Handouts with an explanation of how a transaction transfers funds.

12.3 How a web page gets rendered.

1. URL
2. URL to IP Address
3. IP to server
4. Server looks up file or makes calls to code
5. Different types of calls (GET, POST, PUT, DELETE, OPTIONS)
6. What is HTTPS - how it works
7. What is an API, a RESTful API
8. What is “full stack”.

13 Redo of Transactions and start on Full Stack

13.1 Review of what we did last time

13.2 News

1. IBM and Walmart are requiring food tractability (IBM Food Trust) by next September on all Walmart food.
2. Europe/Russia/China are moving to a non-SWIFT financial system due to Iran Sanctions. Europe Finally Has an Excuse to Challenge the Dollar
3. Secure and anatomized DNA data - India is using blockchian to keep private DNA data in program to find genetic causes of Type 2 Diabetes. Type 2 Diabetes Epidemic in India Combatted with use of Blockchain. This is using an ERC-721 contract on Ethereum.

13.3 Back to the question of investment

?Apartments?

13.4 Software Economics

13.4.1 Why software companies make gobs of money.

13.5 Blockchain Economics

1. Currency - Sub-Saharan Africa
2. Supply Chain - Merick Shipping
3. Raising Capital - ICO, Industrial Staker Sale, Scams
4. Proof of Authenticity -
5. Anonomization and privacy.

13.6 Paper for Midterm

3-5 Pages. Economics of Blockchain.
Examples:

1. “How blockchain is changing the beef industry”.
2. “Blockchain for the Art Market - Proof of Authenticity.”
3. “Tracking Real Estate / Titles / Mechanics Leans and Blockchain.”
4. “How blockchain will change supply chains.”
5. “Blockchain for Central Banking - Provable bank reserve currency.”
6. “Blockchain for Accounting - Then end of Auditing.”

==========

13.7 News

1. ICO Bitman in Hong Cong https://techcrunch.com/2018/09/26/bitmain-hong-kong-ipo/
2. Ripple $100 Million Philanthropy Fund, “Ripple for Good”
3. Blockchain casino - FunFair

13.8 Impostor Feeling

1. Office hours, Tu Noon, Th 5:00 pm.
2. By appointment
3. Getting some interesting comments in Email

13.9 Blockchain in NGOs/Non-Profits/503(c)

1. Proof of trust
2. Tracking of donations

Salvation Army in Haiti 150000 houses, $500 million raised, 3 built.
Only 37% of Americans trust that non-profit donations really go to intended beneficiaries.
PAC's seem to be the worst.

13.10 Smart Contracts

```
contract GetID {

    uint256 protected id;
    event IdGenerated(address sender, uint256 Id);

    constructor() public {
        id = 1;
    }

    function getNextID() public view returns(address, uint256) {
        id = id + 1;
        event IdGenerated(this.sender, id);
        return(this.sender, id);
    }

}
```

13.11 Paper for Midterm

3-5 Pages. Economics of Blockchain. How is it being used now? How could you apply a proof of authenticity or a distributed trust system to a particular part of the economy. How much would it be worth?
Examples:

1. "How blockchain is changing the beef industry".
2. "Blockchain for the Art Market - Proof of Authenticity."
3. "Tracking Real Estate / Titles / Mechanics Leans and Blockchain."
4. "How blockchain will change supply chains."
5. "Blockchain for Central Banking - Provable bank reserve currency."
6. "Blockchain for Accounting - Then end of Auditing."

14 Blockchain and Accounting, Land Ownership, Titles, Signatures

14.1 News

1. Sierra Leone's president Julius Maada Bio announced a new initiative to create a national, blockchain-based credit bureau. The announcement took place at the 73rd Session of the U.N. General Assembly (UNGA) Thursday, September 27. … The initiative is reportedly the result of a partnership between technology non-profit Kiva, the U.N. Capital Development Fund (UNCDF), and the U.N. Development Programme (UNDP). The new system is intended to tackle two major barriers that many citizens face when they attempt to access financial services – these being a lack of both formal identification and a verifiable credit history. As the press release outlines:

"Currently, unbanked people cannot leverage financial transactions from the 'informal economy,' such as credit with a local shopkeeper, to build their credit history. The Kiva Protocol will capture a wide range of financial transactions … to help people access the financial services they need, including loans for businesses, education or basic medical services."

2. Mongolia: Central Bank Gives Permission to Issue First Digital Currency. Mongolia's largest mobile telecoms operator has become the country's first licensed entity to issue its own digital currency. Mobicom's financial arm Mobifinance is now clear to issue the e-currency, dubbed "Candy," …

3. The Prime Minister of Malta, Joseph Muscat, has said that cryptocurrencies are the "inevitable future of money," and that blockchain can galvanize a more transparent and equitable society. The Prime Minister made his remarks in a speech addressed to the general debate of the 73rd Session of the General Assembly of the U.N. September 27.

"Blockchain makes cryptocurrencies the inevitable future of money, more transparent since it helps filter good businesses from bad businesses. But these distributed ledger technologies can do much more."

14.2 Accounting - Double Entry

Basics of Double Entry accounting.
From wikipdia.com: "The oldest European record of a complete double-entry system is the Messari (Italian: Treasurer's) accounts of the Republic of Genoa in 1340. The Messari accounts contain debits and credits journalised in a bilateral form, and include balances carried forward from the preceding year, and therefore enjoy general recognition as a double-entry system."
Some indication this may be from Korea around the year 926

14.3 When was Zero Invented

From the 13th century, manuals on calculation (adding, multiplying, extracting roots, etc.) became common in Europe where they were called algorismus after the Persian mathematician al-Khwārizmī. The most popular was written by Johannes de Sacrobosco, about 1235 and was one of the earliest scientific books to be printed in 1488. Until the late 15th century, Hindu–Arabic numerals seem to have predominated among mathematicians, while merchants preferred to use the Roman numerals. In the 16th century, they became commonly used in Europe.

14.4 Where were negative numbers invented.

Western mathematicians accepted the idea of negative numbers by the 17th century. Prior to the concept of negative numbers, mathematicians such as Diophantus considered negative solutions to problems "false" and equations requiring negative solutions were described as absurd.

14.5 Debits and Credos

```
Debits      | Credits
------------|-----------
Assets      | Liabilities
Draw        | Equity
Expense     | Revinue
```

Debits side equals Credits side.
Example

1. A Loan from the SBA for $50,000.00. Asset+Liabilities.

```
Debits      | Credits
------------|-----------
Assets      | Liabilities
Cash:50000    | Loan:50000
```

2. Rent for $1000. An Expense.

```
Debits      | Credits
------------|-----------
Expense     |
Rent:1000    | Cash:1000
```

2. CNC Milling Machine for $4000. An Asset.

```
Debits      | Credits
------------|-----------
Asset         |
CNC:4000    | Cash:4000
```

3. Money to owner of business $500. A “draw”.

```
Debits      | Credits
------------|-----------
Draw         |
Owner:500    | Cash:500
```

4. Services $12,000.00.

```
Debits      | Credits
------------|-----------
Assets         |
Cash:12000    | Service:12,000
```

5. Intangible Assets - fair value. There is nothing micky-mouse about micky-mouse.

14.6 Cooking the books

1. Suspense Account, 10% kickback.
2. Paul Manfort (Counts #4, #5 - convicted).
3. Overtime Hours, 10 laborers with overtime billed to company.
4. Labor Theft, $33 billion in labor theft a year.
5. Plugging in values (URS Griner Woodward Clyde), $5,000,000.00.
6. Cash and my Auto Mechanic / Airplane Mechanic / Boat Mechanic. Tax Evasion.

15 Introduction to Ethereum Development

15.1 News

1. UN reports that over 1/2 of the world is now “middle class”.

2. Wyoming Bills - Upcoming.

    1. Banking Bill
    2. Digital Signatures
    3. Trusts

3. IBM Patent NO: 10,091,228

15.2 A note from last time

"Blockchain is the first technological innovation in accounting in over 500 years."

15.3 Current Wyoming Bills

(See Law Review Article)

15.4 Software to Install

1. npm Download and install `node.js` from https://nodejs.org/en/download/
2. truffle You should be able to:

```
npm install -g truffle
```

3. ganache

```
npm install -g ganache-cli
```

4. solc (solidity)

Just for entertainment there is a 'solc' compiler and a 'solc-js' compiler. The 'solc-js' compile has different options. We will want to use the 'solc' compiler.
On Windows the general consensus is to setup the Linux Subsystem for Windows. [https://medium.com/@m_mcclarty/setting up-solidity-on-windows-10-993a1d2c615c](https://medium.com/@m_mcclarty/setting-up-solidity-on-windows-10-993a1d2c615c)

15.4.1 Or for Windows

https://github.com/ethereum/solidity/releases
Download the Windows binary from https://github.com/ethereum/solidity/releases
Extract the `solidity-windows.zip` into a new folder.
Launch a command prompt and cd into the directory where `solc.exe` was extracted to.
Move `solc.exe` to a suitable directory to run it form. Usually I have a `bin` directory in my loign home with a path set in the environment that includes `bin`.

15.4.2 On Mac

```
brew update
brew upgrade
brew tap ethereum/ethereum
brew install solidity
```

Notes: 1. https://truffleframework.com/docs/truffle/getting-started/installation

```
\$ truffle develop
Truffle Develop started at http://127.0.0.1:9545/

Accounts:
(0) 0x627306090abab3a6e1400e9345bc60c78a8bef57
(1) 0xf17f52151ebef6c7334fad080c5704d77216b732
(2) 0xc5fdf4076b8f3a5357c5e395ab970b5b54098fef
(3) 0x821aea9a577a9b44299b9c15c88cf3087f3b5544
(4) 0x0d1d4e623d10f9fba5db95830f7d3839406c6af2
(5) 0x2932b7a2355d6fecc4b5c0b6bd44cc31df247a2e
(6) 0x2191ef87e392377ec08e7c08eb105ef5448eced5
```

```
(7) 0x0f4f2ac550a1b4e2280d04c21cea7ebd822934b5
(8) 0x6330a553fc93768f612722bb8c2ec78ac90b3bbc
(9) 0x5aeda56215b167893e80b4fe645ba6d5bab767de

Private Keys:
(0) c87509a1c067bbde78beb793e6fa76530b6382a4c0241e5e4a9ec0a0f44dc0d3
(1) ae6ae8e5ccbfb04590405997ee2d52d2b330726137b875053c36d94e974d162f
(2) 0dbbe8e4ae425a6d2687f1a7e3ba17bc98c673636790f1b8ad91193c05875ef1
(3) c88b703fb08cbea894b6aeff5a544fb92e78a18e19814cd85da83b71f772aa6c
(4) 388c684f0ba1ef5017716adb5d21a053ea8e90277d0868337519f97bede61418
(5) 659cbb0e2411a44db63778987b1e22153c086a95eb6b18bdf89de078917abc63
(6) 82d052c865f5763aad42add438569276c00d3d88a2d062d36b2bae914d58b8c8
(7) aa3680d5d48a8283413f7a108367c7299ca73f553735860a87b08f39395618b7
(8) 0f62d96d6675f32685bbdb8ac13cda7c23436f63efbb9d07700d8669ff12b7c4
(9) 8d5366123cb560bb606379f90a0bfd4769eecc0557f1b362dcae9012b548b1e5

Mnemonic: candy maple cake sugar pudding cream honey rich smooth crumble sweet treat

⚠  Important ⚠  : This mnemonic was created for you by Truffle. It is not secure.
Ensure you do not use it on production blockchains, or else you risk losing funds.

truffle(develop)>
```

15.5 Creating Wealth (if time available)

1. The fixed size "pie" fallacy.
2. What a "startup" really is.

[]@ll@

Description
$101,000.00 Average CoSci Earnings2x Work twice as hard.2x Work twice as smart (More knowledge of what you are working on)3x Pick useful stuff to work on.2x No pointy hard boss.$2,424,000.00 What you should be paid.

16 Wallets

16.1 News

1. I will be gone Oct 9th through 12th. You still have class on 10th. No office hours next week - call me - we can do zoom and remote work but it will be a odd hours in the evening.
2. Wyoming News: Beef blockchain buzz... https://www.tsln.com/news/beef-blockchain-buzz-technology-adds-teeth-to-

16.2 Best analogy I have heard for what is "blockchain"

Cross word puzzles

16.3 What is a wallet

What is in it
Who owns it
Fallacies

16.4 Client/Server - and Homework 5

Server - what is up with that?
Create a "genesis" block
Can run stuff on CLI still [this is the server]
Offline Usage of data
Run the Server

16.5 Where and what to do in the code -

Step 1 - get it to compile

```
Fix the go environment to build this --

We have one build problem that we have to work around.
```

```
\$ cd ~/go/src/github.com/ethereum/go-ethereum
\$ cd vendor/github.com/
```

Remove the pborman directory.

```
\$ rm -rf ./pborman
```

Now do the same with our vendor copy of it.

```
\$ cd ~/go/src/github.com/Univ-Wyo-Education/Blockchain-4010-Fall-2018
\$ cd vendor/github.com
\$ rm -rf ./pborman
```

Now go and get a common copy

```
\$ cd ~/go/src/github.com/
\$ mkdir pborman
\$ cd pborman
\$ git clone https://github.com/pborman/uuid.git
```

Or update your copy you already have with

```
\$ cd ~/go/src/github.com/pborman/uuid
\$ git pull
```

16.6 Step 2 - create genesis block

You have to get the main to compile. In `.../A-05/bsvr/cli/svr-cli.go` around line 116 comment out the call to my implementation - and replace it with `return true, nil` so that it will validate all signatures.

```
\$ cd ../main
\$ go build
\$ ./main --create-genesis
```

16.7 Step 3 - List accounts - locally

```
\$ ./main --list-accts
```

16.8 Step 4 - Run Server

```
\$ ./main --server 127.0.0.1:9000
```

Mac

```
\$ ifconfig
\$ ./main --server 192.168.0.157:9000
```

PC

```
C:\ ipconfig
C:\ main.exe --server 10.27.4.44:9000
```

16.9 Step 5 - compile the client

In the `.../A-05/wallet-client` compile and run the client.

```
\$ cd wallet-client
\$ go build
\$ ./wallet-client --cmd echo
```

16.10 Step 6 - List accounts - remote

```
\$ ./wallet-client --list-accts
```

16.11 Step 7 - Look at command line code

In `.../A-05/sig-test/` look at `*.go`. There are 2 files to pay attention to and a 3rd file to pull some cod out of. First off. You are implementing verification of signatures. There is a `verifyMessage.go`. It has code in it you can adapt to run in the server. There is a `signMessage.go` this has code in it you can adapt to your client. Lastly you will need to pull a function out of `utils.go` - probably just copy the function `func signHash(data []byte) []byte {` and put that into your code. It is used by both the signature and the validation process.

16.12 Step 8 - Implement the signature verification process.

This is in `.../A-05/bsvr/cli/svr-cli.go` around line 116. (The place where you had to comment out the call the signature validation and return a constant true)
Remember that you also have to implement the server side. Step 2 just returned true.

16.13 Step 9 - Implement the signature signing process.

Implement 2 functions in `.../A-05/wallet-client/client-main.go` around lines 110 and 114. The one at 110 is just a signature test. Do that first. This will test the sign/verify process on both sides and that you have client/server communication down.

16.14 Step 10 - run and test the client making calls to the server.

Implement transfer of funds. (The stuff on line 114 of client-main.go). This sends a message with a send-funds to the server.

17 Standard Contracts (ERC-20)

17.1 News

1.
2.
3.
4.

17.2 Standard ERC-20 Contract

17.2.1 SimpleToken

[]@llll@

Method Name	Const	$	Params
Approval	event		(address owner, address spender, uint256 value)
INITIAL_SUPPLY	const		() returns (uint256)
Transfer	event		(address from, address to, uint256 value)
allowance	const		(address _owner, ad
Tx			(address _spender, uint256 _value) returns (bool)
balanceOf	const		(address _owner) returns (uint256)
decimals	const		() returns (uint8)
Tx			(address _spender, uint256 _subtractedValue) returns (bool)
increaseApproval	Tx		(address _spender, uint256 _addedValue) returns (bool)
name	const		() returns (string)
const			() returns (string)
totalSupply	const		() returns (uint256)
transfer	Tx		(address _to, uint25
Tx			(address _from, address _to, uint256 _value) returns (bool)
constructor			()

17.2.2 SimpleToken Ours derived from StandardToken

```
pragma solidity ^0.4.24;

import "openzeppelin-solidity/contracts/token/ERC20/StandardToken.sol";

/**
 * @title SimpleToken
 * @dev Very simple ERC20 Token example, where all tokens are pre-assigned to the creator.
 * Note they can later distribute these tokens as they wish using `transfer` and other
 * `StandardToken` functions.
 */
contract SimpleToken is StandardToken {

    string public constant name = "SimpleToken"; // solium-disable-line uppercase
    string public constant symbol = "SIM"; // solium-disable-line uppercase
    uint8 public constant decimals = 0; // solium-disable-line uppercase

    uint256 public constant INITIAL_SUPPLY = 10000 * (10 ** uint256(decimals));

    /**
     * @dev Constructor that gives msg.sender all of existing tokens.
     */
    constructor() public {
        totalSupply_ = INITIAL_SUPPLY;
```

```
        balances[msg.sender] = INITIAL_SUPPLY;
        emit Transfer(0x0, msg.sender, INITIAL_SUPPLY);
    }

}
```

17.2.3 StandardToken

```
pragma solidity ^0.4.24;

import "./BasicToken.sol";
import "./ERC20.sol";

/**
 * @title Standard ERC20 token
 *
 * @dev Implementation of the basic standard token.
 * https://github.com/ethereum/EIPs/issues/20
 * Based on code by FirstBlood: https://github.com/Firstbloodio/token/blob/master/smart_contract/FirstBloodToken.
 */
contract StandardToken is ERC20, BasicToken {

  mapping (address => mapping (address => uint256)) internal allowed;

  /**
   * @dev Transfer tokens from one address to another
   * @param _from address The address which you want to send tokens from
   * @param _to address The address which you want to transfer to
   * @param _value uint256 the amount of tokens to be transferred
   */
  function transferFrom(
    address _from,
    address _to,
    uint256 _value
  )
    public
    returns (bool)
  {
    require(_to != address(0));
    require(_value <= balances[_from]);
    require(_value <= allowed[_from][msg.sender]);

    balances[_from] = balances[_from].sub(_value);
    balances[_to] = balances[_to].add(_value);
    allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
    emit Transfer(_from, _to, _value);
    return true;
  }

  /**
   * @dev Approve the passed address to spend the specified amount of tokens on behalf of msg.sender.
   * Beware that changing an allowance with this method brings the risk that someone may use both the old
   * and the new allowance by unfortunate transaction ordering. One possible solution to mitigate this
   * race condition is to first reduce the spender's allowance to 0 and set the desired value afterwards:
```

```
 * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
 * @param _spender The address which will spend the funds.
 * @param _value The amount of tokens to be spent.
 */
function approve(address _spender, uint256 _value) public returns (bool) {
  allowed[msg.sender][_spender] = _value;
  emit Approval(msg.sender, _spender, _value);
  return true;
}

/**
 * @dev Function to check the amount of tokens that an owner allowed to a spender.
 * @param _owner address The address which owns the funds.
 * @param _spender address The address which will spend the funds.
 * @return A uint256 specifying the amount of tokens still available for the spender.
 */
function allowance(
  address _owner,
  address _spender
 )
  public
  view
  returns (uint256)
{
  return allowed[_owner][_spender];
}

/**
 * @dev Increase the amount of tokens that an owner allowed to a spender.
 * approve should be called when allowed[_spender] == 0. To increment
 * allowed value is better to use this function to avoid 2 calls (and wait until
 * the first transaction is mined)
 * From MonolithDAO Token.sol
 * @param _spender The address which will spend the funds.
 * @param _addedValue The amount of tokens to increase the allowance by.
 */
function increaseApproval(
  address _spender,
  uint256 _addedValue
)
  public
  returns (bool)
{
  allowed[msg.sender][_spender] = (
    allowed[msg.sender][_spender].add(_addedValue));
  emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
  return true;
}

/**
 * @dev Decrease the amount of tokens that an owner allowed to a spender.
 * approve should be called when allowed[_spender] == 0. To decrement
 * allowed value is better to use this function to avoid 2 calls (and wait until
 * the first transaction is mined)
```

```
   * From MonolithDAO Token.sol
   * @param _spender The address which will spend the funds.
   * @param _subtractedValue The amount of tokens to decrease the allowance by.
   */
  function decreaseApproval(
    address _spender,
    uint256 _subtractedValue
  )
    public
    returns (bool)
  {
    uint256 oldValue = allowed[msg.sender][_spender];
    if (_subtractedValue > oldValue) {
      allowed[msg.sender][_spender] = 0;
    } else {
      allowed[msg.sender][_spender] = oldValue.sub(_subtractedValue);
    }
    emit Approval(msg.sender, _spender, allowed[msg.sender][_spender]);
    return true;
  }

}
```

17.2.4 BasicToken

```
pragma solidity ^0.4.24;

import "./ERC20Basic.sol";
import "../../math/SafeMath.sol";

/**
 * @title Basic token
 * @dev Basic version of StandardToken, with no allowances.
 */
contract BasicToken is ERC20Basic {
  using SafeMath for uint256;

  mapping(address => uint256) balances;

  uint256 totalSupply_;

  /**
  * @dev Total number of tokens in existence
  */
  function totalSupply() public view returns (uint256) {
    return totalSupply_;
  }

  /**
  * @dev Transfer token for a specified address
  * @param _to The address to transfer to.
  * @param _value The amount to be transferred.
  */
```

```
  function transfer(address _to, uint256 _value) public returns (bool) {
    require(_to != address(0));
    require(_value <= balances[msg.sender]);

    balances[msg.sender] = balances[msg.sender].sub(_value);
    balances[_to] = balances[_to].add(_value);
    emit Transfer(msg.sender, _to, _value);
    return true;
  }

  /**
  * @dev Gets the balance of the specified address.
  * @param _owner The address to query the the balance of.
  * @return An uint256 representing the amount owned by the passed address.
  */
  function balanceOf(address _owner) public view returns (uint256) {
    return balances[_owner];
  }

}
```

17.2.5 ERC20

```
pragma solidity ^0.4.24;

import "./ERC20Basic.sol";

/**
 * @title ERC20 interface
 * @dev see https://github.com/ethereum/EIPs/issues/20
 */
contract ERC20 is ERC20Basic {
  function allowance(address owner, address spender)
    public view returns (uint256);

  function transferFrom(address from, address to, uint256 value)
    public returns (bool);

  function approve(address spender, uint256 value) public returns (bool);
  event Approval(
    address indexed owner,
    address indexed spender,
    uint256 value
  );
}
```

18 Lecture 18 - Midterm missed some stuff

18.1 News

1. (class News) - Midterm Monday 29th, Paper due by end of month.
2. SFBW - Definity - 7000 transactions a second, last year 300, this year 2000.
3. SEC crack down on non-exempted ICOs.

4. Ethereum Constantinople Delayed.
5. Midterm Grades - from Homework.
    - Grading in class

18.2 Go Closures

1. Go Closures

See: https://gobyexample.com/closures

```
package main

import (
    "fmt"
)

func NewSeq() func() int {
    seq := 0

    return func() int {
        seq++
        return seq
    }
}

func main() {
    SeqA := NewSeq()
    SeqB := NewSeq()

    fmt.Printf("A = %d\n", SeqA())
    fmt.Printf("A = %d\n", SeqA())
    fmt.Printf("A = %d\n", SeqA())
    fmt.Printf("\n")
    fmt.Printf("B = %d\n", SeqB())
    fmt.Printf("B = %d\n", SeqB())
    fmt.Printf("\n")
    fmt.Printf("A = %d\n", SeqA())

}
```

In our use case (Routing of HTTP Requests in Go):

```
bsvr/main/main.go: 145

    http.HandleFunc("/api/validate-signed-message", getRespHandlerValidateSignedMessage(cc))

    ...

type HandlerFunc func(www http.ResponseWriter, req *http.Request)

    ...

func getRespHandlerValidateSignedMessage(cc *cli.CLI) HandlerFunc {
    return func(www http.ResponseWriter, req *http.Request) {
        www.Header().Set("Content-Type", "application/json")
```