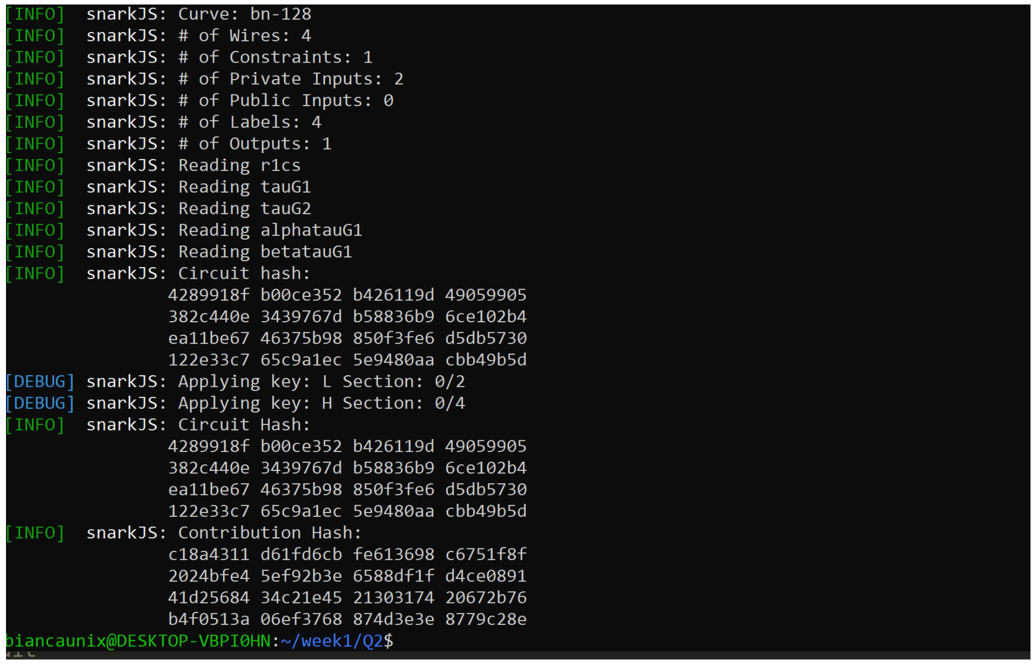
Email: biancabuzea200@gmail.com
Discord: bianca buzea#8157
Github : https://github.com/biancabuzea200/week1

Part 1
1.1 Snark proofs examples: Groth16, PLONK, Spartan
1.2 SNARK(Succint Non-interactive ARgument of Knowledge) requires a trusted setup while STARK (Scalable Transparent of ARgument of Knowledge) doesn't.
SNARKS depend on pairings, whereas STARKS do not depend on pairings. When depending on pairings, a trusted setup is needed. A trusted setup refers to the event of the initial creation of the keys to create the proofs needed for private transactions and for the verification of those proofs. When the keys are being created, there is a hidden parameter linked between the verification key and the keys sending private transactions.

1.3
-SNARKs are estimated to require only 24% of the gas that STARKs would require =>cheaper to use
-proof of size of SNARKs is smaller than that of STARKs =>less on0chain storage
-SNARKs reply on Elliptic Curve cryptography (not resistant to quantum attacks), STARKs rely on hash functions(quantum attacks resistant)

Part 2
2.1 done
2.2.1 The circuit HelloWorld checks that c is the multiplication of a and b.



```
[INFO]  snarkJS: Curve: bn-128
[INFO]  snarkJS: # of Wires: 4
[INFO]  snarkJS: # of Constraints: 1
[INFO]  snarkJS: # of Private Inputs: 2
[INFO]  snarkJS: # of Public Inputs: 0
[INFO]  snarkJS: # of Labels: 4
[INFO]  snarkJS: # of Outputs: 1
[INFO]  snarkJS: Reading r1cs
[INFO]  snarkJS: Reading tauG1
[INFO]  snarkJS: Reading tauG2
[INFO]  snarkJS: Reading alphatauG1
[INFO]  snarkJS: Reading betatauG1
[INFO]  snarkJS: Circuit hash:
                4289918f b00ce352 b426119d 49059905
                382c440e 3439767d b58836b9 6ce102b4
                ea11be67 46375b98 850f3fe6 d5db5730
                122e33c7 65c9a1ec 5e9480aa cbb49b5d
[DEBUG] snarkJS: Applying key: L Section: 0/2
[DEBUG] snarkJS: Applying key: H Section: 0/4
[INFO]  snarkJS: Circuit Hash:
                4289918f b00ce352 b426119d 49059905
                382c440e 3439767d b58836b9 6ce102b4
                ea11be67 46375b98 850f3fe6 d5db5730
                122e33c7 65c9a1ec 5e9480aa cbb49b5d
[INFO]  snarkJS: Contribution Hash:
                c18a4311 d61fd6cb fe613698 c6751f8f
                2024bfe4 5ef92b3e 6588df1f d4ce0891
                41d25684 34c21e45 21303174 20672b76
                b4f0513a 06ef3768 874d3e3e 8779c28e
biancaunix@DESKTOP-VBPI0HN:~/week1/Q2$
```

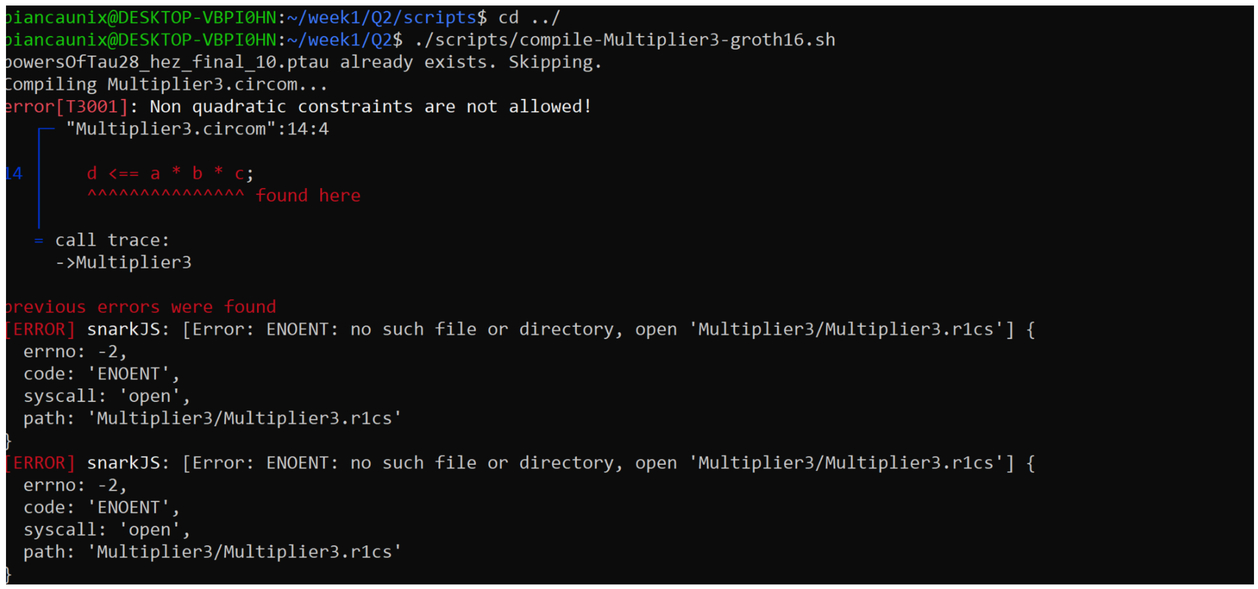
2.2.2 The results of a Powers of Tau ceremony allows for a single, gigantic ceremony to take place for all possible zk-SNARK circuits within a given size bound. The results of this ceremony are partial zk-SNARK parameters for the entire community. We call this communal ceremony the **Powers of Tau**.

2.2.3 Powers of Tau is the first phase from a variant of the MPC ceremony for Groth16 setups. It represents a general setup for all circuits of a given size. The second phase phase convers the output of Powers of Tau phase into a relation-specific CRS.

3.1 done
3.2

Error: “Non quadratic constraints are not allowed". The error refers to the fact that the circuit template must have a function that is maximum quadratic. In this case the function is polynomial, so it returns an error.

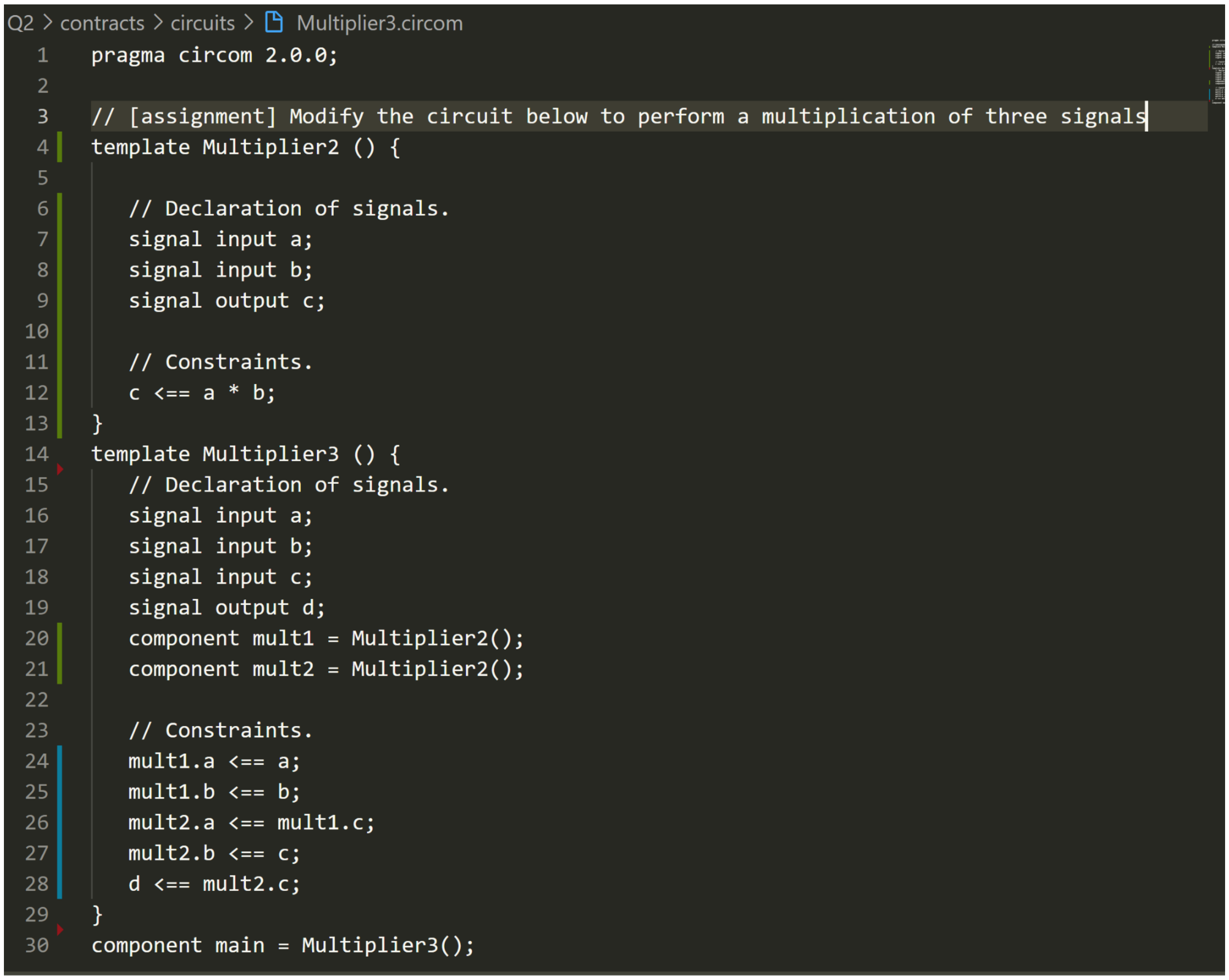
3.3



```
iancaunix@DESKTOP-VBPI0HN:~/week1/Q2/scripts$ cd ../
iancaunix@DESKTOP-VBPI0HN:~/week1/Q2$ ./scripts/compile-Multiplier3-groth16.sh
owersOfTau28_hez_final_10.ptau already exists. Skipping.
ompiling Multiplier3.circom...
rror[T3001]: Non quadratic constraints are not allowed!
   ┌─ "Multiplier3.circom":14:4
   │
4  │     d <== a * b * c;
   │     ^^^^^^^^^^^^^^^ found here
   │
   = call trace:
     ->Multiplier3

revious errors were found
ERROR] snarkJS: [Error: ENOENT: no such file or directory, open 'Multiplier3/Multiplier3.r1cs'] {
  errno: -2,
  code: 'ENOENT',
  syscall: 'open',
  path: 'Multiplier3/Multiplier3.r1cs'
}
ERROR] snarkJS: [Error: ENOENT: no such file or directory, open 'Multiplier3/Multiplier3.r1cs'] {
  errno: -2,
  code: 'ENOENT',
  syscall: 'open',
  path: 'Multiplier3/Multiplier3.r1cs'
}
```

```
pragma circom 2.0.0;

// [assignment] Modify the circuit below to perform a multiplication of three signals

template Multiplier2 () {

   // Declaration of signals.
   signal input a;
   signal input b;
```

```
    signal output c;

    // Constraints.
    c <== a * b;
}

template Multiplier3 () {

    // Declaration of signals.
    signal input a;
    signal input b;
    signal input c;
    signal output d;
    component mult1 = Multiplier2();
    component mult2 = Multiplier2();

    // Constraints.
    d <== a * b * c;
    mult1.a <== a;
    mult1.b <== b;
    mult2.a <== mult1.c;
    mult2.b <== c;
    d <== mult2.c;
}


component main = Multiplier3();
```



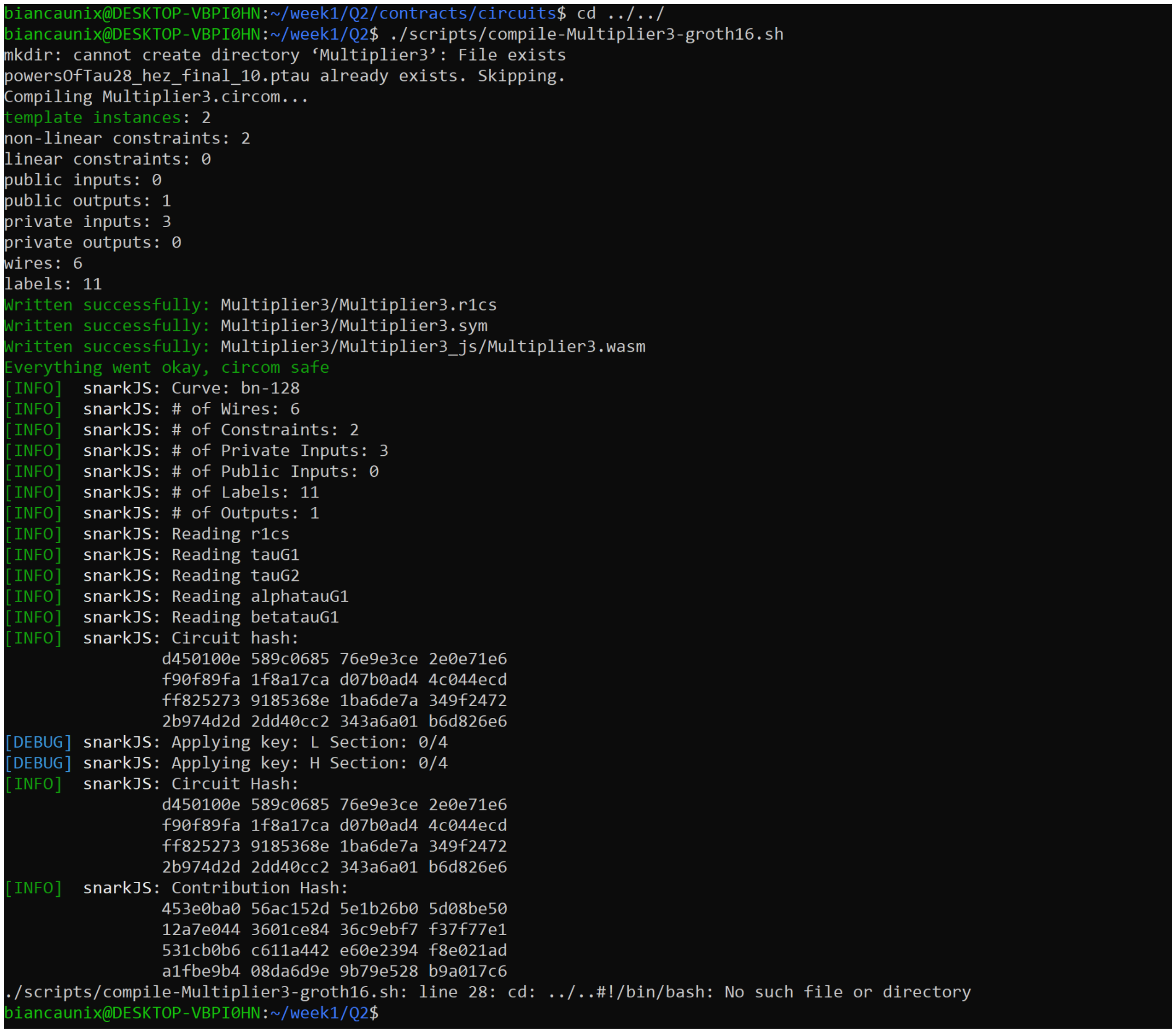
Q2 > contracts > circuits > Multiplier3.circom

```
pragma circom 2.0.0;

// [assignment] Modify the circuit below to perform a multiplication of three signals
template Multiplier2 () {

   // Declaration of signals.
   signal input a;
   signal input b;
   signal output c;

   // Constraints.
   c <== a * b;
}
template Multiplier3 () {
   // Declaration of signals.
   signal input a;
   signal input b;
   signal input c;
   signal output d;
   component mult1 = Multiplier2();
   component mult2 = Multiplier2();

   // Constraints.
   mult1.a <== a;
   mult1.b <== b;
   mult2.a <== mult1.c;
   mult2.b <== c;
   d <== mult2.c;
}
component main = Multiplier3();
```

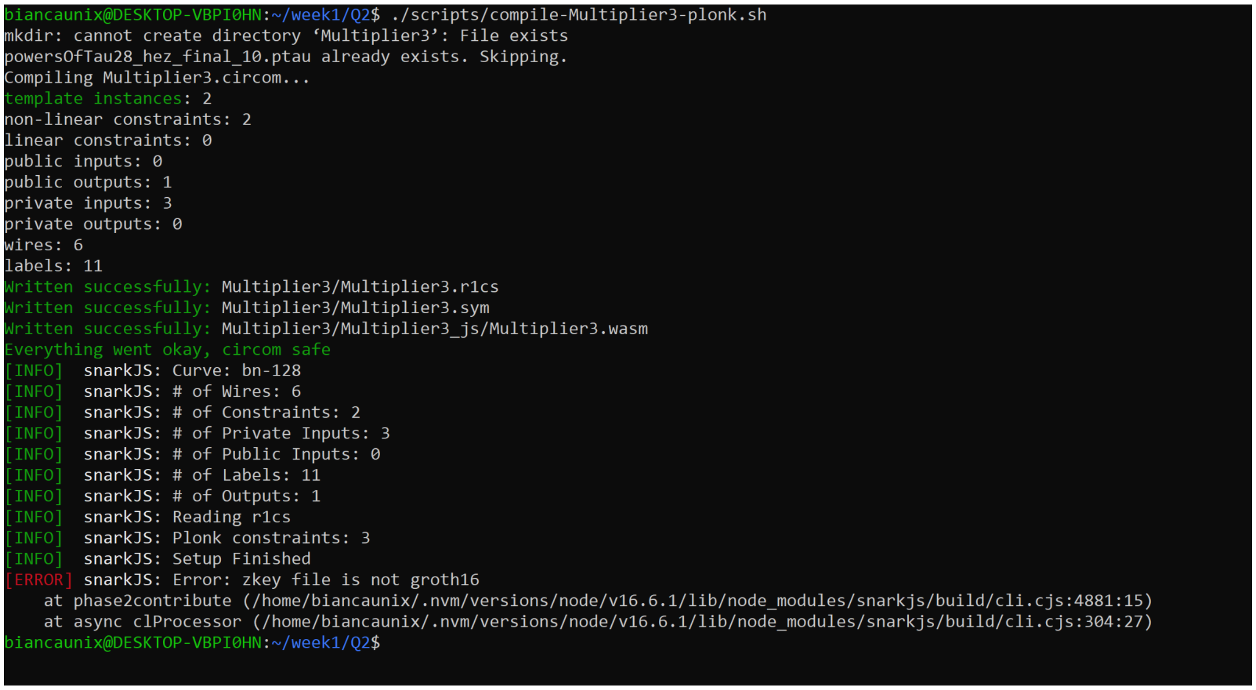
4.

```
#!/bin/bash

# [assignment] create your own bash script to compile Multiplier3.circom
modeling after compile-HelloWorld.sh below

#!/bin/bash

cd contracts/circuits

mkdir Multiplier3

if [ -f ./powersOfTau28_hez_final_10.ptau ]; then
    echo "powersOfTau28_hez_final_10.ptau already exists. Skipping."
else
```

```
    echo 'Downloading powersOfTau28_hez_final_10.ptau'
    wget
https://hermez.s3-eu-west-1.amazonaws.com/powersOfTau28_hez_final_10.ptau
fi

echo "Compiling Multiplier3.circom..."

# compile circuit

circom Multiplier3.circom --r1cs --wasm --sym -o Multiplier3
snarkjs r1cs info Multiplier3/Multiplier3.r1cs

# Start a new zkey and make a contribution

snarkjs groth16 setup Multiplier3/Multiplier3.r1cs
powersOfTau28_hez_final_10.ptau Multiplier3/circuit_0000.zkey
snarkjs zkey contribute Multiplier3/circuit_0000.zkey
Multiplier3/circuit_final.zkey --name="1st Contributor Name" -v -e="random
text"
snarkjs zkey export verificationkey Multiplier3/circuit_final.zkey
Multiplier3/verification_key.json

# generate solidity contract
snarkjs zkey export solidityverifier Multiplier3/circuit_final.zkey
../Multiplier3Verifier.sol

cd ../..
```



```
biancaunix@DESKTOP-VBPI0HN:~/week1/Q2/contracts/circuits$ cd ../../
biancaunix@DESKTOP-VBPI0HN:~/week1/Q2$ ./scripts/compile-Multiplier3-groth16.sh
mkdir: cannot create directory ‘Multiplier3’: File exists
powersOfTau28_hez_final_10.ptau already exists. Skipping.
Compiling Multiplier3.circom...
template instances: 2
non-linear constraints: 2
linear constraints: 0
public inputs: 0
public outputs: 1
private inputs: 3
private outputs: 0
wires: 6
labels: 11
Written successfully: Multiplier3/Multiplier3.r1cs
Written successfully: Multiplier3/Multiplier3.sym
Written successfully: Multiplier3/Multiplier3_js/Multiplier3.wasm
Everything went okay, circom safe
[INFO]  snarkJS: Curve: bn-128
[INFO]  snarkJS: # of Wires: 6
[INFO]  snarkJS: # of Constraints: 2
[INFO]  snarkJS: # of Private Inputs: 3
[INFO]  snarkJS: # of Public Inputs: 0
[INFO]  snarkJS: # of Labels: 11
[INFO]  snarkJS: # of Outputs: 1
[INFO]  snarkJS: Reading r1cs
[INFO]  snarkJS: Reading tauG1
[INFO]  snarkJS: Reading tauG2
[INFO]  snarkJS: Reading alphatauG1
[INFO]  snarkJS: Reading betatauG1
[INFO]  snarkJS: Circuit hash:
                d450100e 589c0685 76e9e3ce 2e0e71e6
                f90f89fa 1f8a17ca d07b0ad4 4c044ecd
                ff825273 9185368e 1ba6de7a 349f2472
                2b974d2d 2dd40cc2 343a6a01 b6d826e6
[DEBUG] snarkJS: Applying key: L Section: 0/4
[DEBUG] snarkJS: Applying key: H Section: 0/4
[INFO]  snarkJS: Circuit Hash:
                d450100e 589c0685 76e9e3ce 2e0e71e6
                f90f89fa 1f8a17ca d07b0ad4 4c044ecd
                ff825273 9185368e 1ba6de7a 349f2472
                2b974d2d 2dd40cc2 343a6a01 b6d826e6
[INFO]  snarkJS: Contribution Hash:
                453e0ba0 56ac152d 5e1b26b0 5d08be50
                12a7e044 3601ce84 36c9ebf7 f37f77e1
                531cb0b6 c611a442 e60e2394 f8e021ad
                a1fbe9b4 08da6d9e 9b79e528 b9a017c6
./scripts/compile-Multiplier3-groth16.sh: line 28: cd: ../..#!/bin/bash: No such file or directory
biancaunix@DESKTOP-VBPI0HN:~/week1/Q2$
```

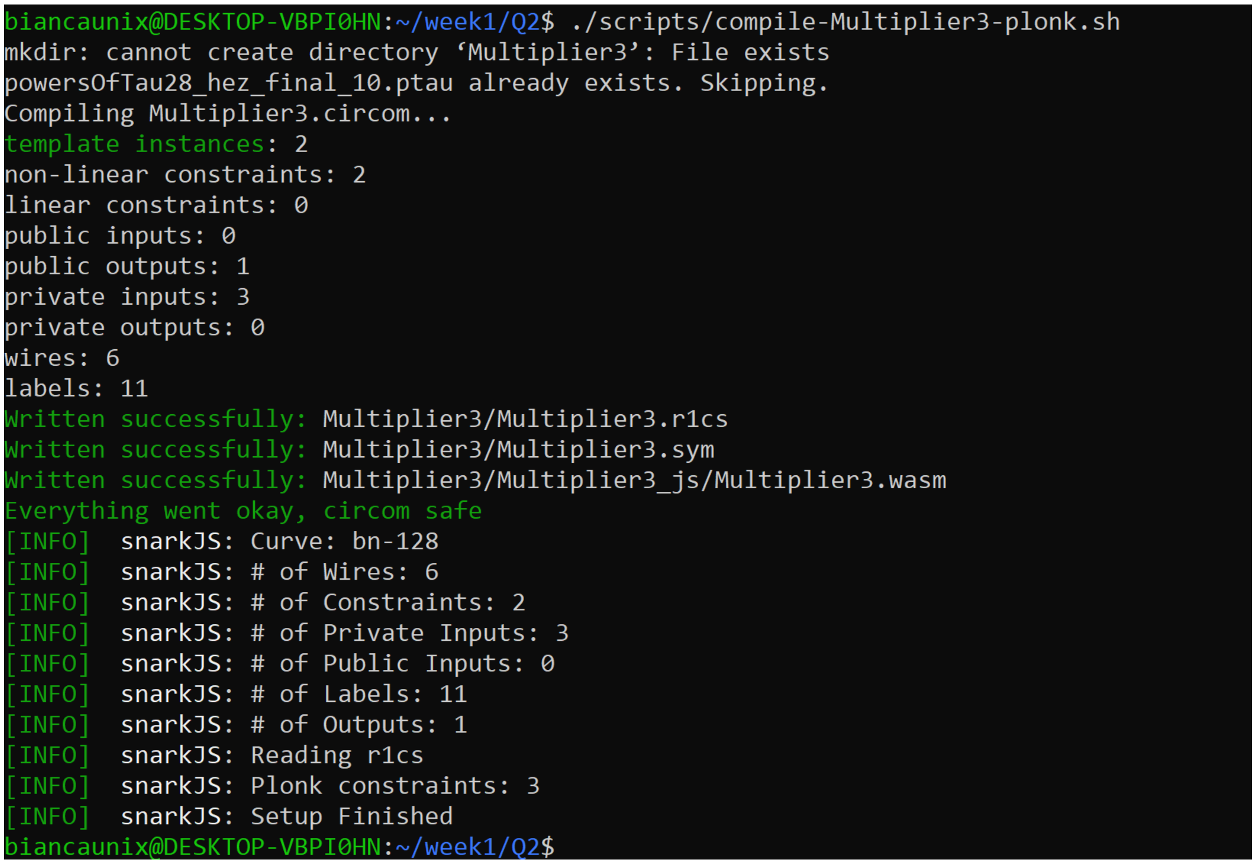


```
biancaunix@DESKTOP-VBPI0HN:~/week1/Q2$ ./scripts/compile-Multiplier3-plonk.sh
mkdir: cannot create directory ‘Multiplier3’: File exists
powersOfTau28_hez_final_10.ptau already exists. Skipping.
Compiling Multiplier3.circom...
template instances: 2
non-linear constraints: 2
linear constraints: 0
public inputs: 0
public outputs: 1
private inputs: 3
private outputs: 0
wires: 6
labels: 11
Written successfully: Multiplier3/Multiplier3.r1cs
Written successfully: Multiplier3/Multiplier3.sym
Written successfully: Multiplier3/Multiplier3_js/Multiplier3.wasm
Everything went okay, circom safe
[INFO]  snarkJS: Curve: bn-128
[INFO]  snarkJS: # of Wires: 6
[INFO]  snarkJS: # of Constraints: 2
[INFO]  snarkJS: # of Private Inputs: 3
[INFO]  snarkJS: # of Public Inputs: 0
[INFO]  snarkJS: # of Labels: 11
[INFO]  snarkJS: # of Outputs: 1
[INFO]  snarkJS: Reading r1cs
[INFO]  snarkJS: Plonk constraints: 3
[INFO]  snarkJS: Setup Finished
[ERROR] snarkJS: Error: zkey file is not groth16
    at phase2contribute (/home/biancaunix/.nvm/versions/node/v16.6.1/lib/node_modules/snarkjs/build/cli.cjs:4881:15)
    at async clProcessor (/home/biancaunix/.nvm/versions/node/v16.6.1/lib/node_modules/snarkjs/build/cli.cjs:304:27)
biancaunix@DESKTOP-VBPI0HN:~/week1/Q2$
```

4.1 PLONK has an universal SNARK which is capable of verifying any arithmetic circuit in one setup. In contrast, Groth16 needs for each circuit a new piece of randomness. That is why when we compiled with PLONK, we had to remove the line for generating the randomness for the specific circuit.

```
d contracts/circuits

mkdir Multiplier3

if [ -f ./powersOfTau28_hez_final_10.ptau ]; then
    echo "powersOfTau28_hez_final_10.ptau already exists. Skipping."
else
    echo 'Downloading powersOfTau28_hez_final_10.ptau'
    wget
https://hermez.s3-eu-west-1.amazonaws.com/powersOfTau28_hez_final_10.ptau
fi

echo "Compiling Multiplier3.circom..."

# compile circuit

circom Multiplier3.circom --r1cs --wasm --sym -o Multiplier3
snarkjs r1cs info Multiplier3/Multiplier3.r1cs
```

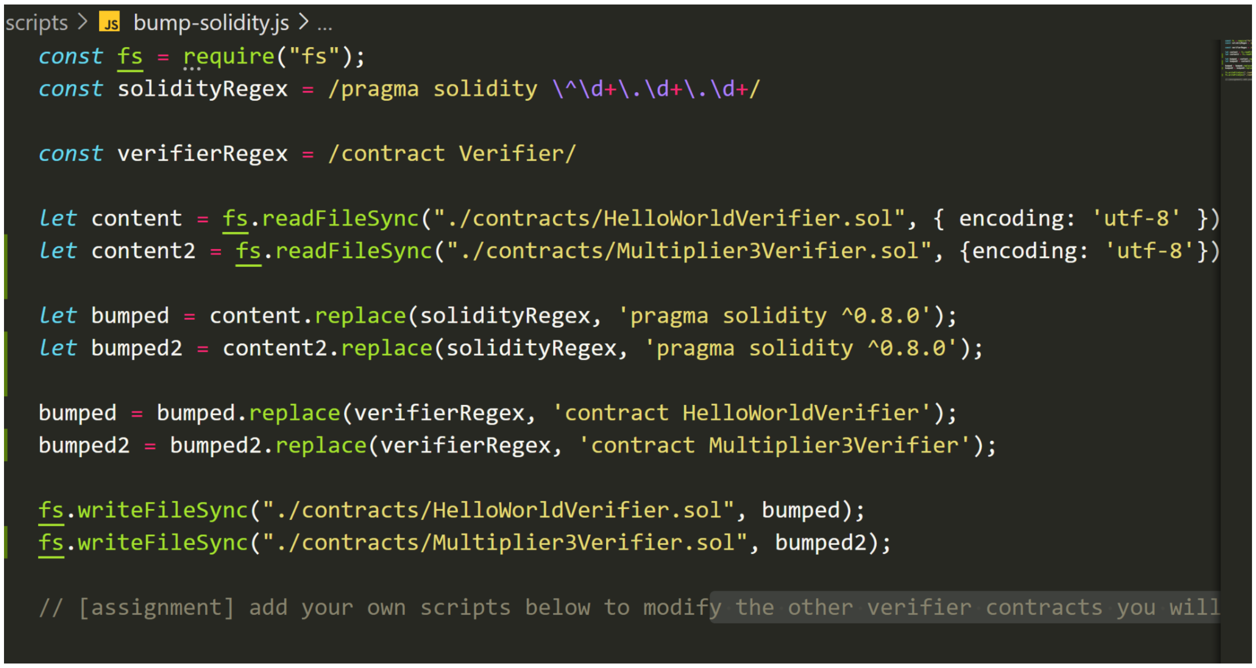
```
# Start a new zkey and make a contribution



snarkjs plonk setup Multiplier3/Multiplier3.r1cs
powersOfTau28_hez_final_10.ptau Multiplier3/circuit_0000.zkey
#snarkjs zkey contribute Multiplier3/circuit_0000.zkey
Multiplier3/circuit_final.zkey --name="1st Contributor Name" -v -e="random
text"
snarkjs zkey export verificationkey Multiplier3/circuit_final.zkey
Multiplier3/verification_key.json

# generate solidity contract
snarkjs zkey export solidityverifier Multiplier3/circuit_final.zkey
../Multiplier3Verifier.sol

cd ../..
```



```
biancaunix@DESKTOP-VBPI0HN:~/week1/Q2$ ./scripts/compile-Multiplier3-plonk.sh
mkdir: cannot create directory ‘Multiplier3’: File exists
powersOfTau28_hez_final_10.ptau already exists. Skipping.
Compiling Multiplier3.circom...
template instances: 2
non-linear constraints: 2
linear constraints: 0
public inputs: 0
public outputs: 1
private inputs: 3
private outputs: 0
wires: 6
labels: 11
Written successfully: Multiplier3/Multiplier3.r1cs
Written successfully: Multiplier3/Multiplier3.sym
Written successfully: Multiplier3/Multiplier3_js/Multiplier3.wasm
Everything went okay, circom safe
[INFO]  snarkJS: Curve: bn-128
[INFO]  snarkJS: # of Wires: 6
[INFO]  snarkJS: # of Constraints: 2
[INFO]  snarkJS: # of Private Inputs: 3
[INFO]  snarkJS: # of Public Inputs: 0
[INFO]  snarkJS: # of Labels: 11
[INFO]  snarkJS: # of Outputs: 1
[INFO]  snarkJS: Reading r1cs
[INFO]  snarkJS: Plonk constraints: 3
[INFO]  snarkJS: Setup Finished
biancaunix@DESKTOP-VBPI0HN:~/week1/Q2$
```

4.2. Running time of unit tests with Groth16 are less than running time of unit tests with PLONK (as it can be seen in Q5). Therefore, PLONK verifies slower than Groth16.

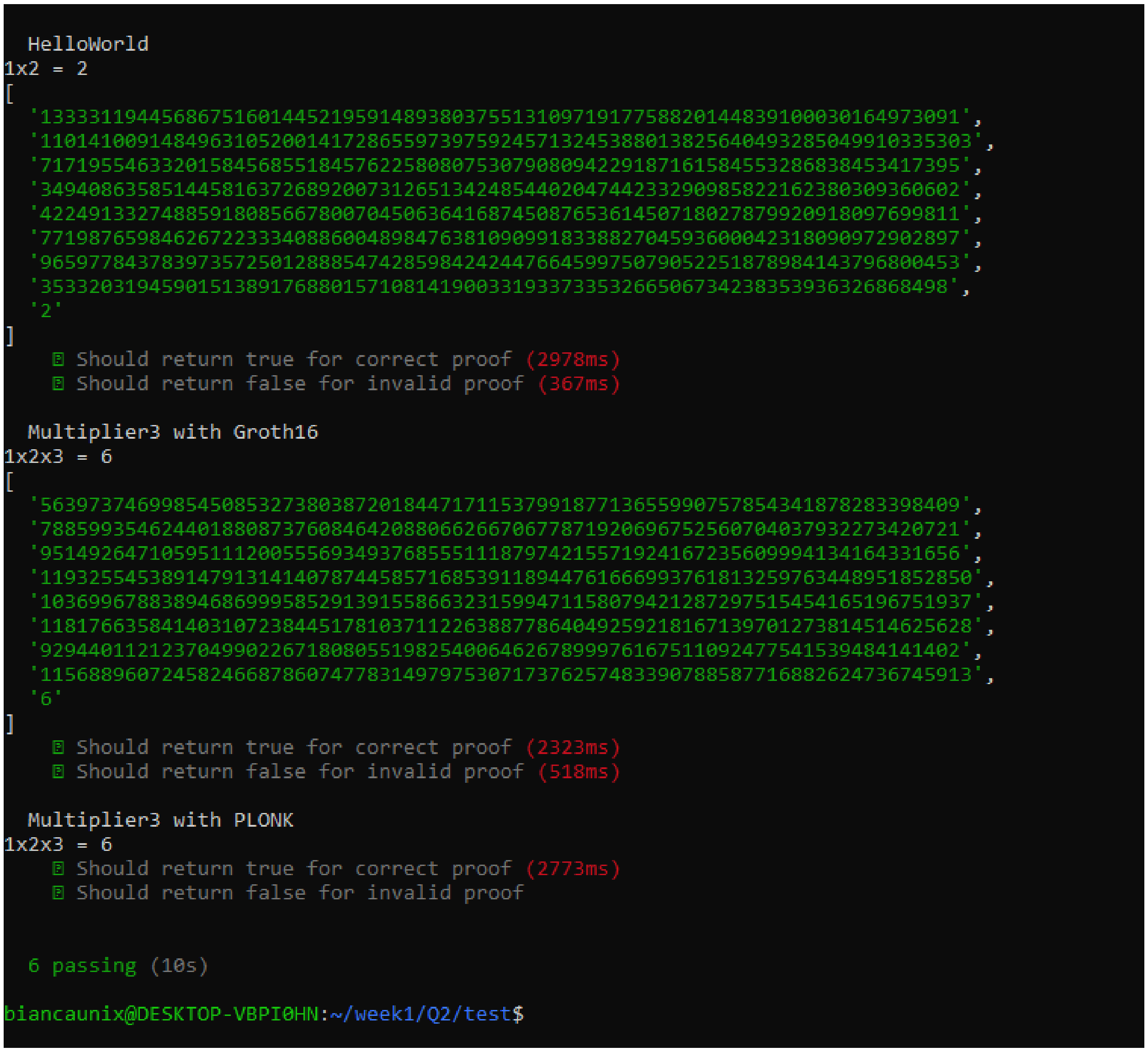
5.1



scripts > bump-solidity.js > ...

```
const fs = require("fs");
const solidityRegex = /pragma solidity \^\d+\.\d+\.\d+/

const verifierRegex = /contract Verifier/

let content = fs.readFileSync("./contracts/HelloWorldVerifier.sol", { encoding: 'utf-8' })
let content2 = fs.readFileSync("./contracts/Multiplier3Verifier.sol", {encoding: 'utf-8'})

let bumped = content.replace(solidityRegex, 'pragma solidity ^0.8.0');
let bumped2 = content2.replace(solidityRegex, 'pragma solidity ^0.8.0');

bumped = bumped.replace(verifierRegex, 'contract HelloWorldVerifier');
bumped2 = bumped2.replace(verifierRegex, 'contract Multiplier3Verifier');

fs.writeFileSync("./contracts/HelloWorldVerifier.sol", bumped);
fs.writeFileSync("./contracts/Multiplier3Verifier.sol", bumped2);

// [assignment] add your own scripts below to modify the other verifier contracts you will
```

5.2

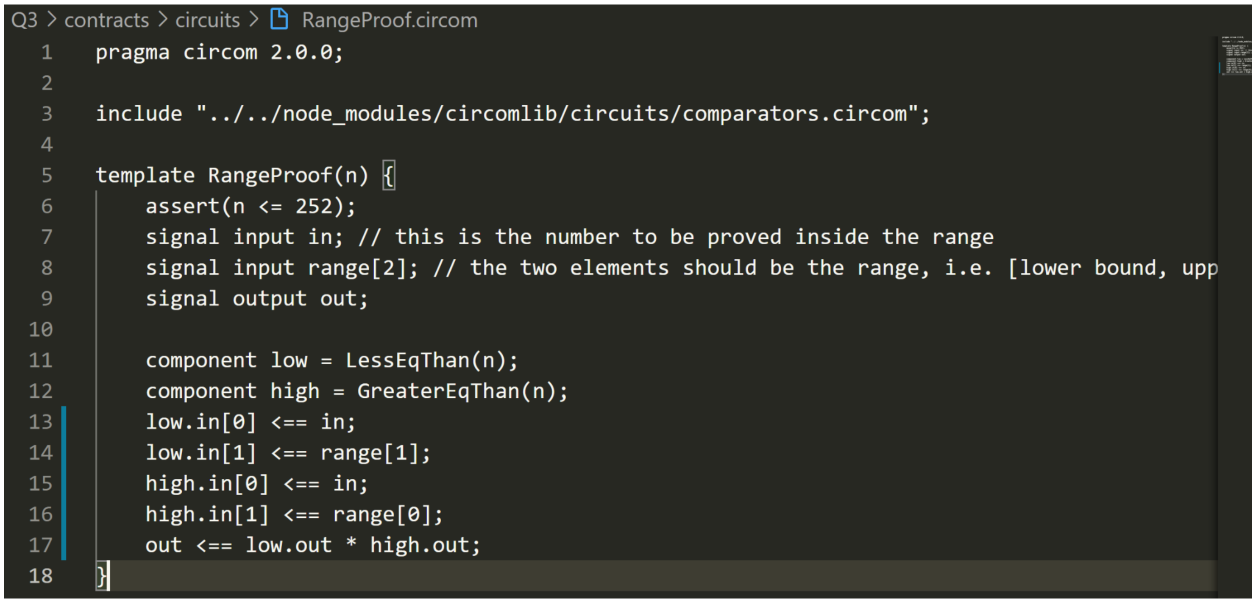


```
  HelloWorld
1x2 = 2
[
  '13333119445686751601445219591489380375513109719177588201448391000301649730 91',
  '11014100914849631052001417286559739759245713245388013825640493285049910335303',
  '7171955463320158456855184576225808075307908094229187161584553286838453417395',
  '3494086358514458163726892007312651342485440204744233290985822162380309360602',
  '4224913327488591808566780070450636416874508765361450718027879920918097699811',
  '7719876598462672233340886004898476381090991833882704593600042318090972902897',
  '9659778437839735725012888547428598424244766459975079052251878984143796800453',
  '3533203194590151389176880157108141900331933733532665067342383539363268684 98',
  '2'
]
    ▣ Should return true for correct proof (2978ms)
    ▣ Should return false for invalid proof (367ms)

  Multiplier3 with Groth16
1x2x3 = 6
[
  '5639737469985450853273803872018447171153799187713655990757854341878283398409',
  '7885993546244018808737608464208806626670677871920696752560704037932273420721',
  '9514926471059511120055569349376855511187974215571924167235609994134164331656',
  '11932554538914791314140787445857168539118944761666993761813259763448951852850',
  '10369967883894686999585291391558663231599471158079421287297515454165196751937',
  '11817663584140310723844517810371122638877864049259218167139701273814514625628',
  '9294401121237049902267180805519825400646267899976167511092477541539484141402',
  '11568896072458246687860747783149797530717376257483390788587716882624736745913',
  '6'
]
    ▣ Should return true for correct proof (2323ms)
    ▣ Should return false for invalid proof (518ms)

  Multiplier3 with PLONK
1x2x3 = 6
    ▣ Should return true for correct proof (2773ms)
    ▣ Should return false for invalid proof


  6 passing (10s)

biancaunix@DESKTOP-VBPI0HN:~/week1/Q2/test$
```

Part3

3.1.1 The 32 (in line 9) stands for the number of bits, therefore 32 bits.

3.1.2 Possible otputs: 0 and 1
3.1.3


Q3 > contracts > circuits > RangeProof.circom

```
pragma circom 2.0.0;

include "../../node_modules/circomlib/circuits/comparators.circom";

template RangeProof(n) {
    assert(n <= 252);
    signal input in; // this is the number to be proved inside the range
    signal input range[2]; // the two elements should be the range, i.e. [lower bound, upp
    signal output out;

    component low = LessEqThan(n);
    component high = GreaterEqThan(n);
    low.in[0] <== in;
    low.in[1] <== range[1];
    high.in[0] <== in;
    high.in[1] <== range[0];
    out <== low.out * high.out;
}
```

3.2.2 Circuit is too big for this power of tau ceremony. This has occurred because power of tau is not large enough for this circuit. To fix this, we need to increase the power of tau.

3.2.4 The logarithmic implementation has a lower complexity than a brute force implementation, therefore it is faster.