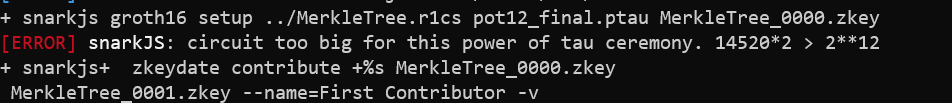
# Assignment 1

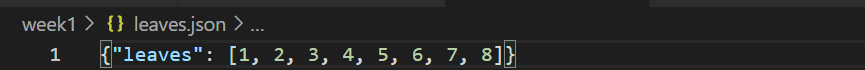
### *Question 1: Intro to circom*

* 1. Construct a circuit using circom that takes a list of numbers input as leaves of a Merkle tree (Note that the numbers will be public inputs) and outputs the Merkle root. For the Merkle hash function, you may use the MiMCsponge hash function from circomlib. For simplicity, you may assume that the number of leaves will be a power of 2 (say 4) and the input will look like this {“leaves”:[1,2,3,4]}

**[Answer]**

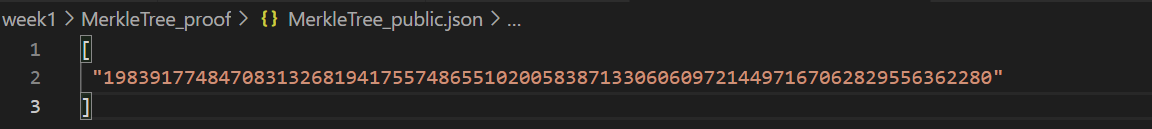
* 1. Now try to generate the proof using a list of 8 numbers. Document any errors (if any) you encounter when increasing the size and explain how you fixed them.  
     **[Answer]** Got error in the script. Change ****
  2. Do we really need zero-knowledge proof for this? Can a publicly verifiable smart contract that computes Merkle root achieve the same? If so, give a scenario where Zero-Knowledge proofs like this might be useful. Are there any technologies implementing this type of proof? Elaborate in 100 words on how they work.  
     **[Answer]**
  3. [Bonus] As you may have noticed, compiling circuits and generating the witness is an elaborate process. Explain what each step is doing. Optionally, you may create a bash script and comment on each step in it. This script will be useful later on to quickly compile circuits.  
     **[Answer]**

Add screenshots of the execution and the generated public.json file.



文本

描述已自动生成



### *Question 2: Minting an NFT and committing the mint data to a Merkle Tree* In this question, we will ask you to start with a simple contract that mints an NFT, then commit the mint data to a Merkle tree contract.

1. Create an ERC721 contract that can mint an NFT to any address. The token URI should be on-chain and should include a name field and a description field. [Bonus points for well-commented codebase]  
   **[Answer]**
2. Commit the msg.sender, receiver address, tokenId, and tokenURI to a Merkle tree using the keccak256 hash function. Update the Merkle tree using a minimal amount of gas.  
   **[Answer]**
3. Use remix to mint a couple of NFTs to the sender address or to other addresses. Include screenshots of the transactions and the amount of gas spent per transaction in your repo.  
   **[Answer]**
4. *[Bonus]* Build a minimal frontend application that allows MetaMask to interact with your contract and mint an NFT.  
   **[Answer]**
5. *[Extra bonus if you did Part (4)]* Use your application to fetch the Merkle leaves from the NFT contract, then generate a proof of the root calculation using snarkjs. Submit this proof to the solidity verifier contract generated from Q1.  
   **[Answer]**

### *Question 3: Understanding and generating ideas about ZK technologies*

Understanding and generating ideas about ZK technologies

1. Summarize the key differences (in application, not in theory) between SNARKs and STARKs in 100 words.  
   **[Answer]**
2. How is the trusted setup process different between Groth16 and PLONK?  
   **[Answer]**
3. Give an idea of how we can apply ZK to create unique usage for NFTs.  
   **[Answer]**
4. [Bonus] Give a novel idea on how we can apply ZK for Dao Tooling.   
   **[Answer]**