NFT Raffle – The Tip Board Challenge

General program description:

Write an NFT Raffle program that functions like a tip board - the key is that the winning number is set at the creation of the raffle, but hidden from other participants while we draw numbers for their tickets. Your program should store the NFT and a **hashed** value for the Winning Number. It should give a random number for Bob's ticket. It should then reveal the winning number.

The purpose of this program is to demonstrate the Reach Cryptographic Commitment Scheme. Commitment schemes are very important in DApps as they can be used to verify trust and ensure a participants commitment to certain values in the program. It also uses non-network tokens to reinforce syntax rules around them.

There are 3 levels to this program. **You can only submit for one level**. Payouts are denoted for each level – you will only receive payout for the level of program you are submitting. (i.e. submit level 2 get $40 USDC)

We recommend building level 1 first and then moving on to level 2. If you are having trouble with level 2 you can fall back on your level 1 solution for submission.

Building a level 3 solution for each of the Hack Challenges is a great way to grow your Reach project portfolio.

Plagiarism Warning

Plagiarism weakens the community and does nothing to improve your web3 development skills. Submissions that are plagiarized will not be accepted.

These programs attempt to get you thinking about building code solutions from word problems. These should be your own work written from scratch.

If you are stuck, lost, or confused, reach out to us on Discord for assistance.

Program Requirements:

- ➔ Level 1 (**$20 USDC**)
    - ◆ Start with reach init
    - ◆ Use 2 Participants for The NFT Raffle Tip Board
        - Deployer deposits the NFT, sets amount of tickets, passes in a winning number
        - Attacher draws a number

- Use launchToken to create a 1 of 1 NFT
  - The Deployer should be responsible for this
  - Publish theNFT.id and be sure to declare it as a Token in the backend
  - Then have the Deployer pay that NFT into the contract
  - Make sure that Bob's account uses tokenAccept(nftId)
- Have Alice declare a hidden Winning Number
  - Select this number randomly between 1 and numberOfTickets
  - Send this number from the frontend to the backend rsh file
  - Then use the Reach Cryptographic Commitment Scheme to hide the value behind a hash. Publish this hashed value and log it to the frontend
    - This will require you to import ...hasRandom for Alice
  - Use an unknowable to verify this information with the Verification Engine
    - Example
    - This should include the private value and the salt value
- Randomly select a number for Bob
  - Pass this number from the frontend to the backend and publish
- Then reveal the winning number from your hashed value
  - Example
  - Be sure to checkCommitment on the values to verify they haven't changed
- Then determine the outcome
  - If Bobs number matches the Winning Number, transfer the NFT to him
  - If Bobs number doesn't match the Winning Number, transfer the NFT back to Alice
    - There are several ways to do this – I'll leave this up to you!
- Display status messages to the console
  - Log a message when the NFT is being created
  - Log a message when you send the raffle parameters to the backend (NFT.id and numberOfTickets)
  - Display the Winning Number HASH value
  - Show Bob his number after you select it
  - Show the winning number after you reveal it
  - Show each user the outcome
- This should be done in 2 files, index.rsh and index.mjs

➔ Level 2 (**$40 USDC**)

- Allow many users with API
  - Read this guide about different types of consensus transfers in Reach
- Give each user a random number between 1 and numberOfTickets
  - Store this information in a Map
- Draw numbers for users until the numberOfTickets is reached
- Then determine the outcome
  - If the number does not match any Bob, transfer back to Alice

        - If the number matches one of the Bobs, transfer to the correct Bob
    - Create an enumeration for the possible outcomes
        - Use makeEnum for this
        - assert that the outcome is equal to a valid outcome
    - Display status messages to the console
        - Log a message when the NFT is being created
        - Log a message when you send The Raffle information to the backend (`NFT.id` and `numberOfTickets`)
        - Display the Winning Number HASH value
        - Show each Bob their number when you select it
        - Show the winningNumber after you reveal it
        - Display the balance of NFT id in the winners account after transfer
    - This should be done in 2 files, index.rsh and index.mjs

➔ Level 3 (**$80 USDC**)

- Include functionality from previous levels
- Only allow each number to be selected **once** between 1 and `numberOfTickets` until all tickets have been drawn
- Then determine the winner and transfer the NFT to the winner
    - There should always be a winner who is not Alice
- Build out a Frontend GUI
    - Use the framework of your choice
    - This will require as many files as you need

☐ **Checklist**

- ☐ **Does your program function like the General Program Description?**
- ☐ **Did you provide solutions for all of the problems on your program level?**
- ☐ **Are you submitting for just one level?**
- ☐ **My submission is my original work**