Blockchain Assignment-3

Core idea:

Creating a smart contract which will include creating NFT token, carry out auction whose owner will be the art owner for selling the NFT at the highest bid and then reselling the NFT using same auction function whose owner will be the highest bidder of previous auction and transferring some amount to original artist of the NFT after selling and reselling the NFT.

Implementation:

First, we will be creating separate codes (.sol) files for Transfer, Auction and NFT token generation.

Since, NFTs are tokens that we can use to represent ownership of unique items, in "NFT_creation.sol" file we will creating NFT token using ERC721 (ERC721 is a standard for representing ownership of NFT, that is, where each token is unique) within the contract named "NEW_NFT"

```
      // creating a constructor for generating token using ERC721
      constructor() ERC721("NEW_NFT", "NFT") {}
```

Then we will be minting, the process of turning a digital file into a crypto collectible or digital asset on the Ethereum blockchain using "mintNFT" function, where we will assign a unique id to the item and then minting the item and then we will be returning the id of the new item.

```
function mintNFT(address recipient, string memory tokenURI)
public onlyOwner
returns (uint256)
{
        // incrementing tokenIDs
        tokenIds.increment();

        // assigning the current tokenIds to new item_Id
        uint256 new_item_Id = tokenIds.current();
        _mint(recipient, new_item_Id);

        // setting token URI
        _setTokenURI(new_item_Id, tokenURI);

        return new_item_Id;
}
```

Whenever the ownership of the NFT gets changed then we will be transferring the token id to the new owner (the highest bidder), so to implement this we will create a function called "token_transfer" which will be carrying out this task safely.

```
function token_transfer(address from, address to, uint256 ID)
public onlyOwner
```

```
{
        safeTransferFrom(from, to, ID);
}
```

In order to sell the NFT, there will be auction whose owner will be the owner of the gallery. To implement the auction, we will create “auction” function where the Owner address and end time will be fixed.

```
function auction (address owner, uint256 biddingTime)
    public
    {
        // auction owner address
        OwnerAddress = owner;
        // closing time of auction
        end_time = block.timestamp + biddingTime;
    }
```

In order to deal with the transaction part of the auction we will create another function called “biding”, here first we will be checking the validity of address of the bidder, if the bid placed within time (before end time), if bid amount is greater than lowest bid and if the amount is greater than current highest bid, then change the top bidder, top bid and emit “topBidIncreased” event.

```
      top_bidder = msg.sender;
      top_bid = msg.value;
      emit topBidIncreased(msg.sender, msg.value);
```

When the time exceed the end time, then the auction will be closed and then emit “auction_results” event and transfer the top bid amount to the owner of the auction.

In the main function we will import the “NEW_NFT” and “Auction”

```
import { NEW_NFT } from './NFT.sol';
import { Auction } from './Auction.sol';
```

The gallery owner will create NFT token corresponding to his digital art using “generate_NFT” function which will return the token id of the NFT. It will generate a new NFT token using NEW_NFT contract which we created in “NFT.sol”.

```
    function generate_NFT(address _owner, string memory digital_art)
    private
    returns (uint256)
    {
        // generating NFT token
        NEW_NFT NFT_token = new NEW_NFT();
        NFT_token_ID = NFT_token.mintNFT(_owner, digital_art);
        // checking whether NFT_token_ID is null if not, then return it
        require(NFT_token_ID != 0);
        return NFT_token_ID;
    }
```

For selling, the art owner will conduct the auction and then will transfer the NFT token to the highest bidder using “NFT_token_transfer” function.

```
    function NFT_token_transfer(address from, address to, uint256 ID)
    private
    {
        // function for transfering NFT token to from one address to another
(owner -> highest bidder)
        NEW_NFT NFT_token = new NEW_NFT();
        NFT_token.token_transfer(from, to, ID);
    }
```

After successfully carrying out the auction by invoking “NFT_transfer_by_auction” function which returns the highest bidder address, the highest bidder will become the owner of the NFT and will transfer 99% of highest bid to the art owner and 1% of highest bid to the artist as well.

Same process will happen while reselling, the only thing which will change is the owner of the auction (here, it will be the highest bidder of the 1st auction)

```
    function NFT_transfer_by_auction(address NFT_owner, uint256 ID)
    private
    returns (address)
    {
        // transferring the digital asset by auction
        simpleAuction auction = new auction();

        auction.auction(NFT_owner,time_to_bid);

        // running the auction untill the closing time
        while (block.timestamp <= closetime)
        {
            auction.biding();
        }

        // closing the auction
        auction.auctionClose();

        // required fields
        require(topBidder != address(0));
        require(topBidder != NFT_owner);
        require(topBid != 0);
        //require(topBid > lowest_bid);

        // after the above requirement gets fullfilled then transfer the token
to top bidder
        NFT_token_transfer(NFT_owner, topBidder, ID);
        // 1% royalty to the artist
        payable(artist).transfer(topBid*1/100);

        return topBidder;
    }
```

Inside main function we will calling the required functions with appropriate parameters, for generating the NFT token the parameters given will be the art_owner and art.

(NOTE: here we are assuming that artist has already sold his art to an art owner by explicit way)

For selling the owner of the auction will be art_owner so the function “NFT_transfer_by_auction” will take parameter as art_owner and NFT_ID, similarly for reselling the first_owner (highest bidder of 1st auction) will be the owner, Hence the parameters will be first_owner and NFT_ID.

```
public
{
        NEW_NFT_token_ID = generate_NFT(art_owner, art); // calling
generate_NFT function and storing token id in variable

        first_owner = NFT_transfer_by_auction(art_owner, NFT_ID); // first
owner

        second_owner = NFT_transfer_by_auction(first_owner, NFT_ID);  //
second owner
}
```

For implementation on Remix platform,

Compile the files in the below order,

1. NFT_creation.sol
2. Auction.sol
3. main.sol

After successfully compiling the codes then we need to deploy in the above order.

Note: Environment – JavaScript VM

Here is the link for remix which contains all the three files,

Link – https://remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js