University of Edinburgh	*Fall 2025*
Blockchains & Distributed Ledgers	

Smart Contract Programming
Coursework Assignment

(Total points = 100, BDL mark weight = 30%)

Due: Tuesday 11.11.2025, 23:59

Please remember the good scholarly practice requirements of the University regarding work for credit. You can find guidance at the School page: https://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct. This also has links to the relevant University pages.

Implementing Auctions On-chain

In this assignment, you will write your own smart contract for facilitating sealed-bid, second-price auctions, also known as Vickrey auctions.

A sealed-bid auction is a two-stage process in which bidders first submit their bids privately so that no one can see anyone else's bid during the bidding period. After bidding closes, the auction's winner is determined as the bidder with the highest valid bid. A second-price auction awards the item to the highest bidder, but the amount paid is the second-highest valid bid or a stated reserve price if that is higher.

The smart contract should enable the following interaction between a seller and multiple bidders. First, a seller starts an auction by specifying the item to be auctioned (more on that later), a public reserve price (a minimum acceptable price) and a time frame for the bidding phase. During the bidding phase, each bidder submits a bid – the amount of the bid should not be visible to anyone else at this point. Bids are considered valid if they are higher or equal to the reserve price. After the bidding phase ends, the winner is determined as the bidder with the highest valid bid. The amount the winner pays is the second-highest valid bid or the reserve price, whichever is larger. If there is only one valid bid, the winner pays the reserve price. If no bids meet the reserve, there is no sale. Ties for the highest bid are broken deterministically by a rule of your choice (e.g., earlier valid submission wins), but this should be documented.

Your contract must enable fair competition without prematurely leaking bid amounts, and it must allow the auction to be finalized and settled deterministically.

Example: Alice, Bob, Charlie and Daisy each have 1,000 wei in their wallet. Alice starts an auction with a reserve price of 90 wei. Bob, Charlie and Daisy participate as bidders, bidding 95 wei, 120 wei, and 100 wei, respectively. The highest bid is Charlie's 120 wei, and the second-highest is Daisy's 100 wei, so Charlie wins and pays 100 wei (the maximum of the second-highest bid and the 90 wei reserve). At the end of the auction, Charlie receives the auctioned item, and then Alice has 1,100 wei in her wallet, Charlie has 900 wei, while Bob and Daisy still have a balance of 1,000 wei each (not considering transaction fees for the sake of the example).

After an auction ends, anyone should be able to start a new auction on the same contract.

The Auction Item: A Sepolia Test NFT

The items that are going to be auctioned are Non-Fungible Tokens (NFTs), which can be created and transferred using an ERC-721 token contract that we have published on the Sepolia testnet. So, for anyone to act as a seller in an auction, they must first mint an NFT using this contract.

To mint an NFT, you need to interact with the deployed contract. You can do this through the Remix IDE, using the code of the contract and its deployed address. The code of the contract can be found here and its address on Sepolia is 0x1546Bd67237122754D3F0cB761c139f81388b210.

The following steps need to be followed to mint tokens and use them in auctions:

1. Mint a new token by calling the *safeMint* function, providing the URI of a json file with the NFT description (an example URI is this).
2. Make a note of the token id (found in the "output" field of the transaction log or in the Etherscan page of the transaction).
3. Verify ownership of your token by calling the function *ownerOf*, with the relevant token id.
4. When launching an auction as a seller, provide the token id of the NFT (your contract should ensure that the seller is indeed the owner of the listed NFT).
5. After the result of the auction is finalized, the token's ownership should be transferred to the winner of the auction.

You should implement the smart contract and deploy it on Ethereum's testnet, Sepolia. Your contract should be as *secure*, *gas efficient*, and *fair* as possible. Specifically, your contract should: i) implement the auction functionality that is described above; ii) not allow any of the participants to cheat (in other words, you should prevent as many attacks as reasonably possible). After you have guaranteed these two requirements, you should try to make it as efficient and fair as possible and detail the trade-off choices you made.

After deploying your contract, you should engage with at least one other student and act as a seller or bidder on their contract; you may use Piazza to find a partner. Before you engage with a fellow student's smart contract, you should evaluate their code and analyze its features in terms of *security, efficiency,* and *fairness* (see Lectures 3-4).

Submission

You should submit **two files** via Learn (in the same Learn submission) choosing the option "Submit via Gradescope." Marking is done anonymously, so read the submission instructions on Learn carefully and *do not* include your name or student number in any of the submitted files (or the name/number of the fellow student with whom you interacted).

The files that should be uploaded are the following:

First, a solidity file that contains the code of your smart contract. The name of the file should be your exam number (e.g., *B123456.sol*).

Second, a PDF report, named again after your exam number (e.g., *B123456.pdf*). The report should contain:

- A detailed description of the high-level decisions you made for the design of your contract, including (but not limited to):
    - How are the bids kept private during the bidding phase?
    - How is the payment sent to the seller?
    - How is it guaranteed that a seller / bidder cannot cheat?
    - How are ties settled?
    - Are bidders allowed to bid multiple times on the same item or not, and how is this handled?
    - What data type/structures do you use and why?
- A detailed gas evaluation of your implementation, including (but not limited to):
    - The cost of deploying and interacting with your contract.

    - Whether your contract is fair to all parties, including whether some bidder has to pay more gas than others and why.
    - Techniques you used to make the contract cost efficient and fair.
- A thorough list of potential vulnerabilities and hazards that *may* occur in the contract. Provide a detailed analysis of the security mechanisms you use to mitigate such hazards and vulnerabilities.
- A detailed description of the tradeoffs and choices you made, e.g., between security and performance, fairness and efficiency, etc.
- Your analysis of your fellow student's contract (along with relative code snippets of their contract), including (but not limited to):
    - Is their implementation secure/efficient/fair?
    - Any vulnerabilities discovered?
        - How could a participant exploit these vulnerabilities to cheat?
- The address of your contract on Sepolia, as well as your fellow student's contract (in the relevant section) and the transaction hashes of your interactions.
- The code of your contract. *(Note: The contract should be both at the end of your PDF report and submitted as a separate file, as described above.)*

The PDF report, excluding the contract's code, should be at most 10 pages (font size at least 11, margin at least 1 inch all around).

Note on the use of AI: You may use AI tools to improve your submission (e.g. for grammar and clarity). However, the core logic of your code and the analysis in your report must be your own original work. You are fully responsible for understanding and being able to explain every part of your submission. If there is any doubt regarding the originality of your submission, you may be required to attend a brief oral examination to explain your work and demonstrate your understanding.

Marking details

Marking follows the Common Marking Scheme and the intention is that each submission's mark will reflect in which grade of the scheme the submission lies. The marks will be roughly distributed as follows.

Criterion	Weighting	Description
Correctness & spec compliance	10%	Does the contract correctly implement the specified sealed bid second-price auction

		logic? E.g., does it determine the winner and the price in the specified manner?
Security	50%	How well does the implementation prevent common smart contract vulnerabilities and reflect all the security considerations relevant to the auction scheme. E.g., does the contract include an effective mechanism to ensure bids remain confidential? Does it ensure that the seller receives the correct payment? Does it ensure that the winner obtains the item? Does the security analysis in the report identify and address these vulnerabilities and the relevant tradeoffs?
Gas (analysis, efficiency and fairness)	25%	Is the code reasonably gas-efficient, and is the analysis of gas costs in the report thorough? Does the contract's design ensure fairness among participants? The analysis should consider if certain roles (e.g., first bidder vs. last bidder) are treated differently in terms of gas costs. The report should justify the trade-offs made between fairness and efficiency.
Peer interaction	10%	Is the analysis of the fellow student's contract (in terms of security, efficiency and fairness) constructive, detailed, and accurate? Was the on-chain interaction completed successfully?
Code quality & documentation	5%	Is the code well-organized, readable, and easy to understand? Is it appropriately documented in the report and / or through comments?