University of Edinburgh	*Fall 2021-22*
Blockchains & Distributed Ledgers	

Assignment #4 (Total points = 100)

Due: Monday 10.1.2022, 16.30

Imagine you have a friend, who wants to create a decentralized implementation of Chess. However, he has only recently started learning about blockchains, so he asks you to design the game instead.

You should produce a technical specification of the game's design on Ethereum. The specification should be as thorough and easily understandable as possible; a reader should be able to understand all requirements and directly implement it on Ethereum simply by reading your report, without needing any additional input from you. Your report doesn't have to list the rules of chess (you may assume that the reader is familiar with them) and you don't have to *actually* implement the game in Solidity (nor write its exact Solidity code). Regardless of what design choices you end up making (as there may be more than one optimal solution), your report should give a detailed justification of why you made each choice.

Your design should employ good practices regarding:

- readability and easy understanding of your design's architecture
- object-oriented programming and properties of inheritance
- gas efficiency and fairness
- security and correctness in the execution of the game

You should submit a single PDF report, at most 10 pages (font size at least 11, margin at least 1 inch all around), that describes in detail the following (along with any other information you deem necessary to make your specification thorough and understandable - the below is not necessarily an exhaustive list):

- how many contracts your design has (if multiple) and what each does
- what custom data structures each contract should define (and what each does)
- the *public* API of your contract(s), including all functions/variables/events
- how a game starts and ends
- how a player interacts with the game's contract(s) to make a move
- if and how a player is notified about the game's state and moves
- proposals for using design and coding patterns that increase gas fairness and efficiency, as well as possible tradeoffs the coder will need to decide upon
- a *secure* randomized process to choose which player gets to play white in every game
- regarding time limits, you don't have to follow the exact real-world rules (e.g. strictly 3/10/100 minutes per player), but you should design your own timing rules that ensure a game does not run forever; your report should explain your choice in detail, taking into account all types of possible attacks (that we discussed across all lectures)
- a description of which parts of the game (if any) could be performed off-chain, to reduce cost, and how this could be done in a way that retains the trust and security guarantees of an (entirely) on-chain execution