Dutch auction

TLDR: Price descends until some bidder is willing to pay it

A *Dutch auction*, also called an *open-bid descending-price auction* or *clock auction*, is a type of auction in which the price of the offering (good) is initially set to a very high value and then gradually lowered. The first bidder to make a bid instantly wins the offering at the current price. After a set time, the auction may end without a winner if the price never goes low enough to attract a bid.

For example, an in-person Dutch auction might start with the auctioneer asking for $1,000. If there are no bidders, the auctioneer might then ask for $900 and continue to lower by $100 every few moments. Eventually, (say for a price of $500), a bidder will accept and obtain the offering for this last announced price of $500.

You will implement a Dutch auction using Ethereum. This contract will implement the auction of a single offering and will take four parameters at the time of creation:

- the initial price
- the number of blocks the auction will be open for, including the block in which the auction is created. That is, the auction starts immediately.
- the (constant) rate at which the price decrements per block. Remember, in a Dutch auction the price gets cheaper the longer the auction is open for.
- whether or not the auction is in “test mode” enabling the time to be manually overridden.

Once created, any address can submit a bid by calling the `bid()` function for this contract. When a bid is received, the contract calculates the current price by querying the current block number and applying the specified rate of decline in price to the original price. The first bid which sends a quantity greater than or equal to the current price is the winner. Any invalid bids should be refunded immediately. The auction’s creator can call `finalize()` after completion to destroy the contract and collect its funds.

A few important notes:

- You should use `msg.sender` to get the sending address, not `tx.origin`.
- Make sure not to call `block.number` directly (as you typically would) but `getBlockNumber()` as this is needed to ensure the tests will work.