EasyAuction.json

{ "address": "0x0b7fFc1f4AD541A4Ed16b40D8c37f0929158D101", "abi": [{ "inputs": [], "stateMutability": "nonpayable", "type": "constructor" }, { "anonymous": false, "inputs": [{ "indexed": true, "internalType": "uint256", "name": "auctionId", "type": "uint256" }, { "indexed": false, "internalType": "uint96", "name": "soldAuctioningTokens", "type": "uint96" }, { "indexed": false, "internalType": "uint96", "name": "soldBiddingTokens", "type": "uint96" }, { "indexed": false, "internalType": "bytes32", "name": "clearingPriceOrder", "type": "bytes32" }], "name": "AuctionCleared", "type": "event" }, { "anonymous": false, "inputs": [{ "indexed": true, "internalType": "uint256", "name": "auctionId", "type": "uint256" }, { "indexed": true, "internalType": "uint64", "name": "userId", "type": "uint64" }, { "indexed": false, "internalType": "uint96", "name": "buyAmount", "type": "uint96" }, { "indexed": false, "internalType": "uint96", "name": "sellAmount", "type": "uint96" }], "name": "CancellationSellOrder", "type": "event" }, { "anonymous": false, "inputs": [{ "indexed": true, "internalType": "uint256", "name": "auctionId", "type": "uint256" }, { "indexed": true, "internalType": "uint64", "name": "userId", "type": "uint64" }, { "indexed": false, "internalType": "uint96", "name": "buyAmount", "type": "uint96" }, { "indexed": false, "internalType": "uint96", "name": "sellAmount", "type": "uint96" }], "name": "ClaimedFromOrder", "type": "event" }, { "anonymous": false, "inputs": [{ "indexed": true, "internalType": "uint256", "name": "auctionId", "type": "uint256" }, { "indexed": true, "internalType": "contract IERC20", "name": "_auctioningToken", "type": "address" }, { "indexed": true, "internalType": "contract IERC20", "name": "_biddingToken", "type": "address" }, { "indexed": false, "internalType": "uint256", "name": "orderCancellationEndDate", "type": "uint256" }, { "indexed": false, "internalType": "uint256", "name": "auctionEndDate", "type": "uint256" }, { "indexed": false, "internalType": "uint64", "name": "userId", "type": "uint64" }, { "indexed": false, "internalType": "uint96", "name": "_auctionedSellAmount", "type": "uint96" }, { "indexed": false, "internalType": "uint96", "name": "_minBuyAmount", "type": "uint96" }, { "indexed": false, "internalType": "uint256", "name": "minimumBiddingAmountPerOrder", "type": "uint256" }, { "indexed": false, "internalType": "uint256", "name": "minFundingThreshold", "type": "uint256" }, { "indexed": false, "internalType": "address", "name": "allowListContract", "type": "address" }, { "indexed": false, "internalType": "bytes", "name": "allowListData", "type": "bytes" }], "name": "NewAuction", "type": "event" }, { "anonymous": false, "inputs": [{ "indexed": true, "internalType": "uint256", "name": "auctionId", "type": "uint256" }, { "indexed": true, "internalType": "uint64", "name": "userId", "type": "uint64" }, { "indexed": false, "internalType": "uint96", "name": "buyAmount", "type": "uint96" }, { "indexed": false, "internalType": "uint96", "name": "sellAmount", "type": "uint96" }], "name": "NewSellOrder",

"type": "event" }, { "anonymous": false, "inputs": [{ "indexed": true, "internalType": "uint64", "name": "userId", "type": "uint64" }, { "indexed": true, "internalType": "address", "name": "userAddress", "type": "address" }], "name": "NewUser", "type": "event" }, { "anonymous": false, "inputs": [{ "indexed": true, "internalType": "address", "name": "previousOwner", "type": "address" }, { "indexed": true, "internalType": "address", "name": "newOwner", "type": "address" }], "name": "OwnershipTransferred", "type": "event" }, { "anonymous": false, "inputs": [{ "indexed": true, "internalType": "address", "name": "user", "type": "address" }, { "indexed": false, "internalType": "uint64", "name": "userId", "type": "uint64" }], "name": "UserRegistration", "type": "event" }, { "inputs": [], "name": "FEE_DENOMINATOR", "outputs": [{ "internalType": "uint256", "name": "", "type": "uint256" }], "stateMutability": "view", "type": "function" }, { "inputs": [{ "internalType": "uint256", "name": "", "type": "uint256" }], "name": "auctionAccessData", "outputs": [{ "internalType": "bytes", "name": "", "type": "bytes" }], "stateMutability": "view", "type": "function" }, { "inputs": [{ "internalType": "uint256", "name": "", "type": "uint256" }], "name": "auctionAccessManager", "outputs": [{ "internalType": "address", "name": "", "type": "address" }], "stateMutability": "view", "type": "function" }, { "inputs": [], "name": "auctionCounter", "outputs": [{ "internalType": "uint256", "name": "", "type": "uint256" }], "stateMutability": "view", "type": "function" }, { "inputs": [{ "internalType": "uint256", "name": "", "type": "uint256" }], "name": "auctionData", "outputs": [{ "internalType": "contract IERC20", "name": "auctioningToken", "type": "address" }, { "internalType": "contract IERC20", "name": "biddingToken", "type": "address" }, { "internalType": "uint256", "name": "orderCancellationEndDate", "type": "uint256" }, { "internalType": "uint256", "name": "auctionEndDate", "type": "uint256" }, { "internalType": "bytes32", "name": "initialAuctionOrder", "type": "bytes32" }, { "internalType": "uint256", "name": "minimumBiddingAmountPerOrder", "type": "uint256" }, { "internalType": "uint256", "name": "interimSumBidAmount", "type": "uint256" }, { "internalType": "bytes32", "name": "interimOrder", "type": "bytes32" }, { "internalType": "bytes32", "name": "clearingPriceOrder", "type": "bytes32" }, { "internalType": "uint96", "name": "volumeClearingPriceOrder", "type": "uint96" }, { "internalType": "bool", "name": "minFundingThresholdNotReached", "type": "bool" }, { "internalType": "bool", "name": "isAtomicClosureAllowed", "type": "bool" }, { "internalType": "uint256", "name": "feeNumerator", "type": "uint256" }, { "internalType": "uint256", "name": "minFundingThreshold", "type": "uint256" }], "stateMutability": "view", "type": "function" }, { "inputs": [{ "internalType": "uint256", "name": "auctionId", "type": "uint256" }, { "internalType": "bytes32[]", "name": "_sellOrders", "type": "bytes32[]" }], "name": "cancelSellOrders", "outputs": [], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [{ "internalType": "uint256", "name": "auctionId", "type": "uint256" }, { "internalType": "bytes32[]", "name": "orders", "type": "bytes32[]" }], "name": "claimFromParticipantOrder", "outputs": [{ "internalType": "uint256", "name": "sumAuctioningTokenAmount", "type": "uint256" }, { "internalType": "uint256", "name": "sumBiddingTokenAmount", "type": "uint256" }], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [{ "internalType": "uint256", "name": "auctionId", "type": "uint256" }, {

"internalType": "bytes32", "name": "order", "type": "bytes32" }], "name": "containsOrder", "outputs": [{ "internalType": "bool", "name": "", "type": "bool" }], "stateMutability": "view", "type": "function" }, { "inputs": [], "name": "feeNumerator", "outputs": [{ "internalType": "uint256", "name": "", "type": "uint256" }], "stateMutability": "view", "type": "function" }, { "inputs": [], "name": "feeReceiverUserId", "outputs": [{ "internalType": "uint64", "name": "", "type": "uint64" }], "stateMutability": "view", "type": "function" }, { "inputs": [{ "internalType": "uint256", "name": "auctionId", "type": "uint256" }], "name": "getSecondsRemainingInBatch", "outputs": [{ "internalType": "uint256", "name": "", "type": "uint256" }], "stateMutability": "view", "type": "function" }, { "inputs": [{ "internalType": "address", "name": "user", "type": "address" }], "name": "getUserId", "outputs": [{ "internalType": "uint64", "name": "userId", "type": "uint64" }], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [{ "internalType": "contract IERC20", "name": "_auctioningToken", "type": "address" }, { "internalType": "contract IERC20", "name": "_biddingToken", "type": "address" }, { "internalType": "uint256", "name": "orderCancellationEndDate", "type": "uint256" }, { "internalType": "uint256", "name": "auctionEndDate", "type": "uint256" }, { "internalType": "uint96", "name": "_auctionedSellAmount", "type": "uint96" }, { "internalType": "uint96", "name": "_minBuyAmount", "type": "uint96" }, { "internalType": "uint256", "name": "minimumBiddingAmountPerOrder", "type": "uint256" }, { "internalType": "uint256", "name": "minFundingThreshold", "type": "uint256" }, { "internalType": "bool", "name": "isAtomicClosureAllowed", "type": "bool" }, { "internalType": "address", "name": "accessManagerContract", "type": "address" }, { "internalType": "bytes", "name": "accessManagerContractData", "type": "bytes" }], "name": "initiateAuction", "outputs": [{ "internalType": "uint256", "name": "", "type": "uint256" }], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [], "name": "numUsers", "outputs": [{ "internalType": "uint64", "name": "", "type": "uint64" }], "stateMutability": "view", "type": "function" }, { "inputs": [], "name": "owner", "outputs": [{ "internalType": "address", "name": "", "type": "address" }], "stateMutability": "view", "type": "function" }, { "inputs": [{ "internalType": "uint256", "name": "auctionId", "type": "uint256" }, { "internalType": "uint96[]", "name": "_minBuyAmounts", "type": "uint96[]" }, { "internalType": "uint96[]", "name": "_sellAmounts", "type": "uint96[]" }, { "internalType": "bytes32[]", "name": "_prevSellOrders", "type": "bytes32[]" }, { "internalType": "bytes", "name": "allowListCallData", "type": "bytes" }], "name": "placeSellOrders", "outputs": [{ "internalType": "uint64", "name": "userId", "type": "uint64" }], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [{ "internalType": "uint256", "name": "auctionId", "type": "uint256" }, { "internalType": "uint96[]", "name": "_minBuyAmounts", "type": "uint96[]" }, { "internalType": "uint96[]", "name": "_sellAmounts", "type": "uint96[]" }, { "internalType": "bytes32[]", "name": "_prevSellOrders", "type": "bytes32[]" }, { "internalType": "bytes", "name": "allowListCallData", "type": "bytes" }, { "internalType": "address", "name": "orderSubmitter", "type": "address" }], "name": "placeSellOrdersOnBehalf", "outputs": [{ "internalType": "uint64", "name": "userId", "type": "uint64" }], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [{

"internalType": "uint256", "name": "auctionId", "type": "uint256" }, { "internalType": "uint256", "name": "iterationSteps", "type": "uint256" }], "name": "precalculateSellAmountSum", "outputs": [], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [{ "internalType": "address", "name": "user", "type": "address" }], "name": "registerUser", "outputs": [{ "internalType": "uint64", "name": "userId", "type": "uint64" }], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [], "name": "renounceOwnership", "outputs": [], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [{ "internalType": "uint256", "name": "newFeeNumerator", "type": "uint256" }, { "internalType": "address", "name": "newfeeReceiverAddress", "type": "address" }], "name": "setFeeParameters", "outputs": [], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [{ "internalType": "uint256", "name": "auctionId", "type": "uint256" }], "name": "settleAuction", "outputs": [{ "internalType": "bytes32", "name": "clearingOrder", "type": "bytes32" }], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [{ "internalType": "uint256", "name": "auctionId", "type": "uint256" }, { "internalType": "uint96[]", "name": "_minBuyAmount", "type": "uint96[]" }, { "internalType": "uint96[]", "name": "_sellAmount", "type": "uint96[]" }, { "internalType": "bytes32[]", "name": "_prevSellOrder", "type": "bytes32[]" }, { "internalType": "bytes", "name": "allowListCallData", "type": "bytes" }], "name": "settleAuctionAtomically", "outputs": [], "stateMutability": "nonpayable", "type": "function" }, { "inputs": [{ "internalType": "address", "name": "newOwner", "type": "address" }], "name": "transferOwnership", "outputs": [], "stateMutability": "nonpayable", "type": "function" }], "transactionHash": "0x5af5443ba9add113a42b0219ac8f398c383dc5a3684a221fd24c5655b8316931", "receipt": { "to": null, "from": "0x8AB8530AEe3E5eD7Ecf5D7CA69c903Ed04595094", "contractAddress": "0x0b7fFc1f4AD541A4Ed16b40D8c37f0929158D101", "transactionIndex": 1, "gasUsed": "5135197", "logsBloom":
"0x000000000000000000000000000000000000000000000000000000008000000000000000000000
000000000000000000000000000200000000000000000000000000000000000000000000000000000
000000000000000010000000000000000000000000000000000000000000200000000000000000008
000000000000000000001000008020004000000000000000000000000000000000000004000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000
400000000000000000000000000000000000000000000000000000000000000000000000000000200
0000000000000000000000000000000000000000000000000000000000000000000000",
"blockHash":
"0xe8fe856b42e4edacb39380c08de382e7156fc60c0e4c9fa4681624c8c5381da5",
"transactionHash":
"0x5af5443ba9add113a42b0219ac8f398c383dc5a3684a221fd24c5655b8316931", "logs": [{ "transactionIndex": 1, "blockNumber": 15264427, "transactionHash": "0x5af5443ba9add113a42b0219ac8f398c383dc5a3684a221fd24c5655b8316931", "address": "0x0b7fFc1f4AD541A4Ed16b40D8c37f0929158D101", "topics": [
"0x8be0079c531659141344cd1fd0a4f28419497f9722a3daafe3b4186f6b6457e0",

"0x0000000000000000000000000000000000000000000000000000000000000000",
"0x0000000000000000000000008ab8530aee3e5ed7ecf5d7ca69c903ed04595094"], "data":
"0x", "logIndex": 0, "blockHash":
"0xe8fe856b42e4edacb39380c08de382e7156fc60c0e4c9fa4681624c8c5381da5" }],
"blockNumber": 15264427, "cumulativeGasUsed": "5215353", "status": 1, "byzantium": true },
"args": [], "solcInputHash": "ad868d512eafc63430765fc273a2c47b", "metadata": "{\"compiler\":
{\"version\":\"0.6.12+commit.27d51765\"},\"language\":\"Solidity\",\"output\":{\"abi\":[{\"inputs\":
[],\"stateMutability\":\"nonpayable\",\"type\":\"constructor\"},{\"anonymous\":false,\"inputs\":
[{\"indexed\":true,\"internalType\":\"uint256\",\"name\":\"auctionId\",\"type\":\"uint256\"},
{\"indexed\":false,\"internalType\":\"uint96\",\"name\":\"soldAuctioningTokens\",\"type\":\"uint96\"
},{\"indexed\":false,\"internalType\":\"uint96\",\"name\":\"soldBiddingTokens\",\"type\":\"uint96\"},
{\"indexed\":false,\"internalType\":\"bytes32\",\"name\":\"clearingPriceOrder\",\"type\":\"bytes32\
"}],\"name\":\"AuctionCleared\",\"type\":\"event\"},{\"anonymous\":false,\"inputs\":
[{\"indexed\":true,\"internalType\":\"uint256\",\"name\":\"auctionId\",\"type\":\"uint256\"},
{\"indexed\":true,\"internalType\":\"uint64\",\"name\":\"userId\",\"type\":\"uint64\"},
{\"indexed\":false,\"internalType\":\"uint96\",\"name\":\"buyAmount\",\"type\":\"uint96\"},
{\"indexed\":false,\"internalType\":\"uint96\",\"name\":\"sellAmount\",\"type\":\"uint96\"}],\"name\"
:\"CancellationSellOrder\",\"type\":\"event\"},{\"anonymous\":false,\"inputs\":
[{\"indexed\":true,\"internalType\":\"uint256\",\"name\":\"auctionId\",\"type\":\"uint256\"},
{\"indexed\":true,\"internalType\":\"uint64\",\"name\":\"userId\",\"type\":\"uint64\"},
{\"indexed\":false,\"internalType\":\"uint96\",\"name\":\"buyAmount\",\"type\":\"uint96\"},
{\"indexed\":false,\"internalType\":\"uint96\",\"name\":\"sellAmount\",\"type\":\"uint96\"}],\"name\"
:\"ClaimedFromOrder\",\"type\":\"event\"},{\"anonymous\":false,\"inputs\":
[{\"indexed\":true,\"internalType\":\"uint256\",\"name\":\"auctionId\",\"type\":\"uint256\"},
{\"indexed\":true,\"internalType\":\"contract
IERC20\",\"name\":\"_auctioningToken\",\"type\":\"address\"},
{\"indexed\":true,\"internalType\":\"contract
IERC20\",\"name\":\"_biddingToken\",\"type\":\"address\"},
{\"indexed\":false,\"internalType\":\"uint256\",\"name\":\"orderCancellationEndDate\",\"type\":\"ui
nt256\"},
{\"indexed\":false,\"internalType\":\"uint256\",\"name\":\"auctionEndDate\",\"type\":\"uint256\"},
{\"indexed\":false,\"internalType\":\"uint64\",\"name\":\"userId\",\"type\":\"uint64\"},
{\"indexed\":false,\"internalType\":\"uint96\",\"name\":\"_auctionedSellAmount\",\"type\":\"uint96\
"},{\"indexed\":false,\"internalType\":\"uint96\",\"name\":\"_minBuyAmount\",\"type\":\"uint96\"},
{\"indexed\":false,\"internalType\":\"uint256\",\"name\":\"minimumBiddingAmountPerOrder\",\"ty
pe\":\"uint256\"},
{\"indexed\":false,\"internalType\":\"uint256\",\"name\":\"minFundingThreshold\",\"type\":\"uint25
6\"},
{\"indexed\":false,\"internalType\":\"address\",\"name\":\"allowListContract\",\"type\":\"address\"}

,
{\"indexed\":false,\"internalType\":\"bytes\",\"name\":\"allowListData\",\"type\":\"bytes\"}],\"name\
":\"NewAuction\",\"type\":\"event\"},{\"anonymous\":false,\"inputs\":
[{\"indexed\":true,\"internalType\":\"uint256\",\"name\":\"auctionId\",\"type\":\"uint256\"},
{\"indexed\":true,\"internalType\":\"uint64\",\"name\":\"userId\",\"type\":\"uint64\"},
{\"indexed\":false,\"internalType\":\"uint96\",\"name\":\"buyAmount\",\"type\":\"uint96\"},
{\"indexed\":false,\"internalType\":\"uint96\",\"name\":\"sellAmount\",\"type\":\"uint96\"}],\"name\"
:\"NewSellOrder\",\"type\":\"event\"},{\"anonymous\":false,\"inputs\":
[{\"indexed\":true,\"internalType\":\"uint64\",\"name\":\"userId\",\"type\":\"uint64\"},
{\"indexed\":true,\"internalType\":\"address\",\"name\":\"userAddress\",\"type\":\"address\"}],\"na
me\":\"NewUser\",\"type\":\"event\"},{\"anonymous\":false,\"inputs\":
[{\"indexed\":true,\"internalType\":\"address\",\"name\":\"previousOwner\",\"type\":\"address\"},
{\"indexed\":true,\"internalType\":\"address\",\"name\":\"newOwner\",\"type\":\"address\"}],\"nam
e\":\"OwnershipTransferred\",\"type\":\"event\"},{\"anonymous\":false,\"inputs\":
[{\"indexed\":true,\"internalType\":\"address\",\"name\":\"user\",\"type\":\"address\"},
{\"indexed\":false,\"internalType\":\"uint64\",\"name\":\"userId\",\"type\":\"uint64\"}],\"name\":\"Us
erRegistration\",\"type\":\"event\"},{\"inputs\":[],\"name\":\"FEE_DENOMINATOR\",\"outputs\":
[{\"internalType\":\"uint256\",\"name\":\"\",\"type\":\"uint256\"}],\"stateMutability\":\"view\",\"type\":
\"function\"},{\"inputs\":
[{\"internalType\":\"uint256\",\"name\":\"\",\"type\":\"uint256\"}],\"name\":\"auctionAccessData\",\"
outputs\":
[{\"internalType\":\"bytes\",\"name\":\"\",\"type\":\"bytes\"}],\"stateMutability\":\"view\",\"type\":\"fu
nction\"},{\"inputs\":
[{\"internalType\":\"uint256\",\"name\":\"\",\"type\":\"uint256\"}],\"name\":\"auctionAccessManage
r\",\"outputs\":
[{\"internalType\":\"address\",\"name\":\"\",\"type\":\"address\"}],\"stateMutability\":\"view\",\"type\
":\"function\"},{\"inputs\":[],\"name\":\"auctionCounter\",\"outputs\":
[{\"internalType\":\"uint256\",\"name\":\"\",\"type\":\"uint256\"}],\"stateMutability\":\"view\",\"type\":
\"function\"},{\"inputs\":
[{\"internalType\":\"uint256\",\"name\":\"\",\"type\":\"uint256\"}],\"name\":\"auctionData\",\"outputs
\":[{\"internalType\":\"contract IERC20\",\"name\":\"auctioningToken\",\"type\":\"address\"},
{\"internalType\":\"contract IERC20\",\"name\":\"biddingToken\",\"type\":\"address\"},
{\"internalType\":\"uint256\",\"name\":\"orderCancellationEndDate\",\"type\":\"uint256\"},
{\"internalType\":\"uint256\",\"name\":\"auctionEndDate\",\"type\":\"uint256\"},
{\"internalType\":\"bytes32\",\"name\":\"initialAuctionOrder\",\"type\":\"bytes32\"},
{\"internalType\":\"uint256\",\"name\":\"minimumBiddingAmountPerOrder\",\"type\":\"uint256\"},
{\"internalType\":\"uint256\",\"name\":\"interimSumBidAmount\",\"type\":\"uint256\"},
{\"internalType\":\"bytes32\",\"name\":\"interimOrder\",\"type\":\"bytes32\"},
{\"internalType\":\"bytes32\",\"name\":\"clearingPriceOrder\",\"type\":\"bytes32\"},

{\"internalType\":\"uint96\",\"name\":\"volumeClearingPriceOrder\",\"type\":\"uint96\"},
{\"internalType\":\"bool\",\"name\":\"minFundingThresholdNotReached\",\"type\":\"bool\"},
{\"internalType\":\"bool\",\"name\":\"isAtomicClosureAllowed\",\"type\":\"bool\"},
{\"internalType\":\"uint256\",\"name\":\"feeNumerator\",\"type\":\"uint256\"},
{\"internalType\":\"uint256\",\"name\":\"minFundingThreshold\",\"type\":\"uint256\"}],\"stateMutability\":\"view\",\"type\":\"function\"},{\"inputs\":
[{\"internalType\":\"uint256\",\"name\":\"auctionId\",\"type\":\"uint256\"},
{\"internalType\":\"bytes32[]\",\"name\":\"_sellOrders\",\"type\":\"bytes32[]\"}],\"name\":\"cancelSellOrders\",\"outputs\":[],\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"inputs\":
[{\"internalType\":\"uint256\",\"name\":\"auctionId\",\"type\":\"uint256\"},
{\"internalType\":\"bytes32[]\",\"name\":\"orders\",\"type\":\"bytes32[]\"}],\"name\":\"claimFromParticipantOrder\",\"outputs\":
[{\"internalType\":\"uint256\",\"name\":\"sumAuctioningTokenAmount\",\"type\":\"uint256\"},
{\"internalType\":\"uint256\",\"name\":\"sumBiddingTokenAmount\",\"type\":\"uint256\"}],\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"inputs\":
[{\"internalType\":\"uint256\",\"name\":\"auctionId\",\"type\":\"uint256\"},
{\"internalType\":\"bytes32\",\"name\":\"order\",\"type\":\"bytes32\"}],\"name\":\"containsOrder\",\"outputs\":
[{\"internalType\":\"bool\",\"name\":\"\",\"type\":\"bool\"}],\"stateMutability\":\"view\",\"type\":\"function\"},{\"inputs\":[],\"name\":\"feeNumerator\",\"outputs\":
[{\"internalType\":\"uint256\",\"name\":\"\",\"type\":\"uint256\"}],\"stateMutability\":\"view\",\"type\":\"function\"},{\"inputs\":[],\"name\":\"feeReceiverUserId\",\"outputs\":
[{\"internalType\":\"uint64\",\"name\":\"\",\"type\":\"uint64\"}],\"stateMutability\":\"view\",\"type\":\"function\"},{\"inputs\":
[{\"internalType\":\"uint256\",\"name\":\"auctionId\",\"type\":\"uint256\"}],\"name\":\"getSecondsRemainingInBatch\",\"outputs\":
[{\"internalType\":\"uint256\",\"name\":\"\",\"type\":\"uint256\"}],\"stateMutability\":\"view\",\"type\":\"function\"},{\"inputs\":
[{\"internalType\":\"address\",\"name\":\"user\",\"type\":\"address\"}],\"name\":\"getUserId\",\"outputs\":
[{\"internalType\":\"uint64\",\"name\":\"userId\",\"type\":\"uint64\"}],\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"inputs\":[{\"internalType\":\"contract
IERC20\",\"name\":\"_auctioningToken\",\"type\":\"address\"},{\"internalType\":\"contract
IERC20\",\"name\":\"_biddingToken\",\"type\":\"address\"},
{\"internalType\":\"uint256\",\"name\":\"orderCancellationEndDate\",\"type\":\"uint256\"},
{\"internalType\":\"uint256\",\"name\":\"auctionEndDate\",\"type\":\"uint256\"},
{\"internalType\":\"uint96\",\"name\":\"_auctionedSellAmount\",\"type\":\"uint96\"},
{\"internalType\":\"uint96\",\"name\":\"_minBuyAmount\",\"type\":\"uint96\"},
{\"internalType\":\"uint256\",\"name\":\"minimumBiddingAmountPerOrder\",\"type\":\"uint256\"},

{\"internalType\":\"uint256\",\"name\":\"minFundingThreshold\",\"type\":\"uint256\"},
{\"internalType\":\"bool\",\"name\":\"isAtomicClosureAllowed\",\"type\":\"bool\"},
{\"internalType\":\"address\",\"name\":\"accessManagerContract\",\"type\":\"address\"},
{\"internalType\":\"bytes\",\"name\":\"accessManagerContractData\",\"type\":\"bytes\"}],\"name\":
\"initiateAuction\",\"outputs\":
[{\"internalType\":\"uint256\",\"name\":\"\",\"type\":\"uint256\"}],\"stateMutability\":\"nonpayable\",\
"type\":\"function\"},{\"inputs\":[],\"name\":\"numUsers\",\"outputs\":
[{\"internalType\":\"uint64\",\"name\":\"\",\"type\":\"uint64\"}],\"stateMutability\":\"view\",\"type\":\"f
unction\"},{\"inputs\":[],\"name\":\"owner\",\"outputs\":
[{\"internalType\":\"address\",\"name\":\"\",\"type\":\"address\"}],\"stateMutability\":\"view\",\"type\
":\"function\"},{\"inputs\":[{\"internalType\":\"uint256\",\"name\":\"auctionId\",\"type\":\"uint256\"},
{\"internalType\":\"uint96[]\",\"name\":\"_minBuyAmounts\",\"type\":\"uint96[]\"},
{\"internalType\":\"uint96[]\",\"name\":\"_sellAmounts\",\"type\":\"uint96[]\"},
{\"internalType\":\"bytes32[]\",\"name\":\"_prevSellOrders\",\"type\":\"bytes32[]\"},
{\"internalType\":\"bytes\",\"name\":\"allowListCallData\",\"type\":\"bytes\"}],\"name\":\"placeSell
Orders\",\"outputs\":
[{\"internalType\":\"uint64\",\"name\":\"userId\",\"type\":\"uint64\"}],\"stateMutability\":\"nonpayabl
e\",\"type\":\"function\"},{\"inputs\":
[{\"internalType\":\"uint256\",\"name\":\"auctionId\",\"type\":\"uint256\"},
{\"internalType\":\"uint96[]\",\"name\":\"_minBuyAmounts\",\"type\":\"uint96[]\"},
{\"internalType\":\"uint96[]\",\"name\":\"_sellAmounts\",\"type\":\"uint96[]\"},
{\"internalType\":\"bytes32[]\",\"name\":\"_prevSellOrders\",\"type\":\"bytes32[]\"},
{\"internalType\":\"bytes\",\"name\":\"allowListCallData\",\"type\":\"bytes\"},
{\"internalType\":\"address\",\"name\":\"orderSubmitter\",\"type\":\"address\"}],\"name\":\"placeS
ellOrdersOnBehalf\",\"outputs\":
[{\"internalType\":\"uint64\",\"name\":\"userId\",\"type\":\"uint64\"}],\"stateMutability\":\"nonpayabl
e\",\"type\":\"function\"},{\"inputs\":
[{\"internalType\":\"uint256\",\"name\":\"auctionId\",\"type\":\"uint256\"},
{\"internalType\":\"uint256\",\"name\":\"iterationSteps\",\"type\":\"uint256\"}],\"name\":\"precalcul
ateSellAmountSum\",\"outputs\":[],\"stateMutability\":\"nonpayable\",\"type\":\"function\"},
{\"inputs\":
[{\"internalType\":\"address\",\"name\":\"user\",\"type\":\"address\"}],\"name\":\"registerUser\",\"o
utputs\":
[{\"internalType\":\"uint64\",\"name\":\"userId\",\"type\":\"uint64\"}],\"stateMutability\":\"nonpayabl
e\",\"type\":\"function\"},{\"inputs\":[],\"name\":\"renounceOwnership\",\"outputs\":
[],\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"inputs\":
[{\"internalType\":\"uint256\",\"name\":\"newFeeNumerator\",\"type\":\"uint256\"},
{\"internalType\":\"address\",\"name\":\"newfeeReceiverAddress\",\"type\":\"address\"}],\"name\"
:\"setFeeParameters\",\"outputs\":[],\"stateMutability\":\"nonpayable\",\"type\":\"function\"},

{\"inputs\":
[{\"internalType\":\"uint256\",\"name\":\"auctionId\",\"type\":\"uint256\"}],\"name\":\"settleAuction\
",\"outputs\":
[{\"internalType\":\"bytes32\",\"name\":\"clearingOrder\",\"type\":\"bytes32\"}],\"stateMutability\":\
"nonpayable\",\"type\":\"function\"},{\"inputs\":
[{\"internalType\":\"uint256\",\"name\":\"auctionId\",\"type\":\"uint256\"},
{\"internalType\":\"uint96[]\",\"name\":\"_minBuyAmount\",\"type\":\"uint96[]\"},
{\"internalType\":\"uint96[]\",\"name\":\"_sellAmount\",\"type\":\"uint96[]\"},
{\"internalType\":\"bytes32[]\",\"name\":\"_prevSellOrder\",\"type\":\"bytes32[]\"},
{\"internalType\":\"bytes\",\"name\":\"allowListCallData\",\"type\":\"bytes\"}],\"name\":\"settleAucti
onAtomically\",\"outputs\":[],\"stateMutability\":\"nonpayable\",\"type\":\"function\"},{\"inputs\":
[{\"internalType\":\"address\",\"name\":\"newOwner\",\"type\":\"address\"}],\"name\":\"transferOw
nership\",\"outputs\":[],\"stateMutability\":\"nonpayable\",\"type\":\"function\"}],\"devdoc\":
{\"kind\":\"dev\",\"methods\":{\"owner()\":{\"details\":\"Returns the address of the current
owner.\"},\"renounceOwnership()\":{\"details\":\"Leaves the contract without owner. It will not be
possible to call `onlyOwner` functions anymore. Can only be called by the current owner.
NOTE: Renouncing ownership will leave the contract without an owner, thereby removing any
functionality that is only available to the owner.\"},\"transferOwnership(address)\":
{\"details\":\"Transfers ownership of the contract to a new account (`newOwner`). Can only be
called by the current owner.\"}},\"version\":1},\"userdoc\":{\"kind\":\"user\",\"methods\":
{},\"version\":1}},\"settings\":{\"compilationTarget\":
{\"contracts/EasyAuction.sol\":\"EasyAuction\"},\"evmVersion\":\"istanbul\",\"libraries\":
{},\"metadata\":{\"bytecodeHash\":\"ipfs\",\"useLiteralContent\":true},\"optimizer\":
{\"enabled\":false,\"runs\":200},\"remappings\":[]},\"sources\":
{\"@openzeppelin/contracts/GSN/Context.sol\":{\"content\":\"// SPDX-License-Identifier:
MIT\\n\\npragma solidity >=0.6.0 <0.8.0;\\n\\n/*\\n * @dev Provides information about the
current execution context, including the\\n * sender of the transaction and its data. While these
are generally available\\n * via msg.sender and msg.data, they should not be accessed in
such a direct\\n * manner, since when dealing with GSN meta-transactions the account
sending and\\n * paying for execution may not be the actual sender (as far as an application\\n
* is concerned).\\n *\\n * This contract is only required for intermediate, library-like contracts.\\n
*/\\nabstract contract Context {\\n function _msgSender() internal view virtual returns (address
payable) {\\n return msg.sender;\\n }\\n\\n function _msgData() internal view virtual returns
(bytes memory) {\\n this; // silence state mutability warning without generating bytecode - see
https://github.com/ethereum/solidity/issues/2691\\n return msg.data;\\n
}\\n}\\n\",\"keccak256\":\"0x8d3cb350f04ff49cfb10aef08d87f19dcbaecc8027b0bed12f3275cd12
f38cf0\",\"license\":\"MIT\"},\"@openzeppelin/contracts/access/Ownable.sol\":{\"content\":\"//
SPDX-License-Identifier: MIT\\n\\npragma solidity >=0.6.0 <0.8.0;\\n\\nimport
\\\"../GSN/Context.sol\\\";\\n/**\\n * @dev Contract module which provides a basic access

control mechanism, where\\n * there is an account (an owner) that can be granted exclusive access to\\n * specific functions.\\n *\\n * By default, the owner account will be the one that deploys the contract. This\\n * can later be changed with {transferOwnership}.\\n *\\n * This module is used through inheritance. It will make available the modifier\\n * `onlyOwner`, which can be applied to your functions to restrict their use to\\n * the owner.\\n */\\nabstract contract Ownable is Context {\\n address private _owner;\\n\\n event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);\\n\\n /**\\n * @dev Initializes the contract setting the deployer as the initial owner.\\n */\\n constructor () internal {\\n address msgSender = _msgSender();\\n _owner = msgSender;\\n emit OwnershipTransferred(address(0), msgSender);\\n }\\n\\n /**\\n * @dev Returns the address of the current owner.\\n */\\n function owner() public view returns (address) {\\n return _owner;\\n }\\n\\n /**\\n * @dev Throws if called by any account other than the owner.\\n */\\n modifier onlyOwner() {\\n require(_owner == _msgSender(), \\"Ownable: caller is not the owner\\");\\n _;\\n }\\n\\n /**\\n * @dev Leaves the contract without owner. It will not be possible to call\\n * `onlyOwner` functions anymore. Can only be called by the current owner.\\n *\\n * NOTE: Renouncing ownership will leave the contract without an owner,\\n * thereby removing any functionality that is only available to the owner.\\n */\\n function renounceOwnership() public virtual onlyOwner {\\n emit OwnershipTransferred(_owner, address(0));\\n _owner = address(0);\\n }\\n\\n /**\\n * @dev Transfers ownership of the contract to a new account (`newOwner`).\\n * Can only be called by the current owner.\\n */\\n function transferOwnership(address newOwner) public virtual onlyOwner {\\n require(newOwner != address(0), \\"Ownable: new owner is the zero address\\");\\n emit OwnershipTransferred(_owner, newOwner);\\n _owner = newOwner;\\n }\\n}\\n\",\"keccak256\":\"0xf7c39c7e6d06ed3bda90cfefbcbf2ddc32c599c3d6721746546ad64946efccaa\",\"license\":\"MIT\"},\"@openzeppelin/contracts/math/Math.sol\":{\"content\":\"// SPDX-License-Identifier: MIT\\n\\npragma solidity >=0.6.0 <0.8.0;\\n\\n/**\\n * @dev Standard math utilities missing in the Solidity language.\\n */\\nlibrary Math {\\n /**\\n * @dev Returns the largest of two numbers.\\n */\\n function max(uint256 a, uint256 b) internal pure returns (uint256) {\\n return a >= b ? a : b;\\n }\\n\\n /**\\n * @dev Returns the smallest of two numbers.\\n */\\n function min(uint256 a, uint256 b) internal pure returns (uint256) {\\n return a < b ? a : b;\\n }\\n\\n /**\\n * @dev Returns the average of two numbers. The result is rounded towards\\n * zero.\\n */\\n function average(uint256 a, uint256 b) internal pure returns (uint256) {\\n // (a + b) / 2 can overflow, so we distribute\\n return (a / 2) + (b / 2) + ((a % 2 + b % 2) / 2);\\n }\\n}\\n\",\"keccak256\":\"0x363bd3b45201f07c9b71c2edc96533468cf14a3d029fabd82fddceb1eb3ebd9c\",\"license\":\"MIT\"},\"@openzeppelin/contracts/math/SafeMath.sol\":{\"content\":\"// SPDX-License-Identifier: MIT\\n\\npragma solidity >=0.6.0 <0.8.0;\\n\\n/**\\n * @dev Wrappers over Solidity's arithmetic operations with added overflow\\n * checks.\\n *\\n * Arithmetic operations in Solidity wrap on overflow. This can easily result\\n * in bugs, because

programmers usually assume that an overflow raises an\\n * error, which is the standard behavior in high level programming languages.\\n * `SafeMath` restores this intuition by reverting the transaction when an\\n * operation overflows.\\n *\\n * Using this library instead of the unchecked operations eliminates an entire\\n * class of bugs, so it's recommended to use it always.\\n */\\nlibrary SafeMath {\\n /**\\n * @dev Returns the addition of two unsigned integers, reverting on\\n * overflow.\\n *\\n * Counterpart to Solidity's `+` operator.\\n *\\n * Requirements:\\n *\\n * - Addition cannot overflow.\\n */\\n function add(uint256 a, uint256 b) internal pure returns (uint256) {\\n uint256 c = a + b;\\n require(c >= a, \\\"SafeMath: addition overflow\\\");\\n\\n return c;\\n }\\n\\n /**\\n * @dev Returns the subtraction of two unsigned integers, reverting on\\n * overflow (when the result is negative).\\n *\\n * Counterpart to Solidity's `-` operator.\\n *\\n * Requirements:\\n *\\n * - Subtraction cannot overflow.\\n */\\n function sub(uint256 a, uint256 b) internal pure returns (uint256) {\\n return sub(a, b, \\\"SafeMath: subtraction overflow\\\");\\n }\\n\\n /**\\n * @dev Returns the subtraction of two unsigned integers, reverting with custom message on\\n * overflow (when the result is negative).\\n *\\n * Counterpart to Solidity's `-` operator.\\n *\\n * Requirements:\\n *\\n * - Subtraction cannot overflow.\\n */\\n function sub(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {\\n require(b <= a, errorMessage);\\n uint256 c = a - b;\\n\\n return c;\\n }\\n\\n /**\\n * @dev Returns the multiplication of two unsigned integers, reverting on\\n * overflow.\\n *\\n * Counterpart to Solidity's `*` operator.\\n *\\n * Requirements:\\n *\\n * - Multiplication cannot overflow.\\n */\\n function mul(uint256 a, uint256 b) internal pure returns (uint256) {\\n // Gas optimization: this is cheaper than requiring 'a' not being zero, but the\\n // benefit is lost if 'b' is also tested.\\n // See: https://github.com/OpenZeppelin/openzeppelin-contracts/pull/522\\n if (a == 0) {\\n return 0;\\n }\\n\\n uint256 c = a * b;\\n require(c / a == b, \\\"SafeMath: multiplication overflow\\\");\\n\\n return c;\\n }\\n\\n /**\\n * @dev Returns the integer division of two unsigned integers. Reverts on\\n * division by zero. The result is rounded towards zero.\\n *\\n * Counterpart to Solidity's `/` operator. Note: this function uses a\\n * `revert` opcode (which leaves remaining gas untouched) while Solidity\\n * uses an invalid opcode to revert (consuming all remaining gas).\\n *\\n * Requirements:\\n *\\n * - The divisor cannot be zero.\\n */\\n function div(uint256 a, uint256 b) internal pure returns (uint256) {\\n return div(a, b, \\\"SafeMath: division by zero\\\");\\n }\\n\\n /**\\n * @dev Returns the integer division of two unsigned integers. Reverts with custom message on\\n * division by zero. The result is rounded towards zero.\\n *\\n * Counterpart to Solidity's `/` operator. Note: this function uses a\\n * `revert` opcode (which leaves remaining gas untouched) while Solidity\\n * uses an invalid opcode to revert (consuming all remaining gas).\\n *\\n * Requirements:\\n *\\n * - The divisor cannot be zero.\\n */\\n function div(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {\\n require(b > 0, errorMessage);\\n uint256 c = a / b;\\n // assert(a == b * c + a % b); // There is no case in which this doesn't hold\\n\\n return c;\\n }\\n\\n /**\\n * @dev Returns the remainder of dividing two unsigned integers. (unsigned integer modulo),\\n * Reverts when

dividing by zero.\\n *\\n * Counterpart to Solidity's `%` operator. This function uses a `revert`\\n * opcode (which leaves remaining gas untouched) while Solidity uses an\\n * invalid opcode to revert (consuming all remaining gas).\\n *\\n * Requirements:\\n *\\n * - The divisor cannot be zero.\\n */\\n function mod(uint256 a, uint256 b) internal pure returns (uint256) {\\n return mod(a, b, \\\"SafeMath: modulo by zero\\\");\\n }\\n\\n /**\\n * @dev Returns the remainder of dividing two unsigned integers. (unsigned integer modulo),\\n * Reverts with custom message when dividing by zero.\\n *\\n * Counterpart to Solidity's `%` operator. This function uses a `revert`\\n * opcode (which leaves remaining gas untouched) while Solidity uses an\\n * invalid opcode to revert (consuming all remaining gas).\\n *\\n * Requirements:\\n *\\n * - The divisor cannot be zero.\\n */\\n function mod(uint256 a, uint256 b, string memory errorMessage) internal pure returns (uint256) {\\n require(b != 0, errorMessage);\\n return a % b;\\n }\\n}\\n\",\"keccak256\":\"0x3b21f2c8d626de3b9925ae33e972d8bf5c8b1bffb3f4ee94daeed7d0679036e6\",\"license\":\"MIT\"},\"@openzeppelin/contracts/token/ERC20/IERC20.sol\":{\"content\":\"// SPDX-License-Identifier: MIT\\n\\npragma solidity >=0.6.0 <0.8.0;\\n\\n/**\\n * @dev Interface of the ERC20 standard as defined in the EIP.\\n */\\ninterface IERC20 {\\n /**\\n * @dev Returns the amount of tokens in existence.\\n */\\n function totalSupply() external view returns (uint256);\\n\\n /**\\n * @dev Returns the amount of tokens owned by `account`.\\n */\\n function balanceOf(address account) external view returns (uint256);\\n\\n /**\\n * @dev Moves `amount` tokens from the caller's account to `recipient`.\\n *\\n * Returns a boolean value indicating whether the operation succeeded.\\n *\\n * Emits a {Transfer} event.\\n */\\n function transfer(address recipient, uint256 amount) external returns (bool);\\n\\n /**\\n * @dev Returns the remaining number of tokens that `spender` will be\\n * allowed to spend on behalf of `owner` through {transferFrom}. This is\\n * zero by default.\\n *\\n * This value changes when {approve} or {transferFrom} are called.\\n */\\n function allowance(address owner, address spender) external view returns (uint256);\\n\\n /**\\n * @dev Sets `amount` as the allowance of `spender` over the caller's tokens.\\n *\\n * Returns a boolean value indicating whether the operation succeeded.\\n *\\n * IMPORTANT: Beware that changing an allowance with this method brings the risk\\n * that someone may use both the old and the new allowance by unfortunate\\n * transaction ordering. One possible solution to mitigate this race\\n * condition is to first reduce the spender's allowance to 0 and set the\\n * desired value afterwards:\\n * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729\\n *\\n * Emits an {Approval} event.\\n */\\n function approve(address spender, uint256 amount) external returns (bool);\\n\\n /**\\n * @dev Moves `amount` tokens from `sender` to `recipient` using the\\n * allowance mechanism. `amount` is then deducted from the caller's\\n * allowance.\\n *\\n * Returns a boolean value indicating whether the operation succeeded.\\n *\\n * Emits a {Transfer} event.\\n */\\n function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);\\n\\n /**\\n * @dev Emitted when `value` tokens are moved from one account (`from`) to\\n * another (`to`).\\n *\\n * Note that `value` may be zero.\\n */\\n event Transfer(address indexed from, address indexed to, uint256 value);\\n\\n /**\\n * @dev

Emitted when the allowance of a `spender` for an `owner` is set by\\n * a call to {approve}. `value` is the new allowance.\\n */\\n event Approval(address indexed owner, address indexed spender, uint256 value);\\n}\\n\",\"keccak256\":\"0x5f02220344881ce43204ae4a6281145a67bc52c2bb1290a791857df3d19d78f5\",\"license\":\"MIT\"},\"@openzeppelin/contracts/token/ERC20/SafeERC20.sol\":{\"content\":\"// SPDX-License-Identifier: MIT\\n\\npragma solidity >=0.6.0 <0.8.0;\\n\\nimport \\\"./IERC20.sol\\\";\\nimport \\\"../../math/SafeMath.sol\\\";\\nimport \\\"../../utils/Address.sol\\\";\\n\\n/**\\n * @title SafeERC20\\n * @dev Wrappers around ERC20 operations that throw on failure (when the token\\n * contract returns false). Tokens that return no value (and instead revert or\\n * throw on failure) are also supported, non-reverting calls are assumed to be\\n * successful.\\n * To use this library you can add a `using SafeERC20 for IERC20;` statement to your contract,\\n * which allows you to call the safe operations as `token.safeTransfer(...)`, etc.\\n */\\nlibrary SafeERC20 {\\n using SafeMath for uint256;\\n using Address for address;\\n\\n function safeTransfer(IERC20 token, address to, uint256 value) internal {\\n _callOptionalReturn(token, abi.encodeWithSelector(token.transfer.selector, to, value));\\n }\\n\\n function safeTransferFrom(IERC20 token, address from, address to, uint256 value) internal {\\n _callOptionalReturn(token, abi.encodeWithSelector(token.transferFrom.selector, from, to, value));\\n }\\n\\n /**\\n * @dev Deprecated. This function has issues similar to the ones found in\\n * {IERC20-approve}, and its usage is discouraged.\\n *\\n * Whenever possible, use {safeIncreaseAllowance} and\\n * {safeDecreaseAllowance} instead.\\n */\\n function safeApprove(IERC20 token, address spender, uint256 value) internal {\\n // safeApprove should only be called when setting an initial allowance,\\n // or when resetting it to zero. To increase and decrease it, use\\n // 'safeIncreaseAllowance' and 'safeDecreaseAllowance'\\n // solhint-disable-next-line max-line-length\\n require((value == 0) || (token.allowance(address(this), spender) == 0),\\n \\\"SafeERC20: approve from non-zero to non-zero allowance\\\"\\n);\\n _callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender, value));\\n }\\n\\n function safeIncreaseAllowance(IERC20 token, address spender, uint256 value) internal {\\n uint256 newAllowance = token.allowance(address(this), spender).add(value);\\n _callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender, newAllowance));\\n }\\n\\n function safeDecreaseAllowance(IERC20 token, address spender, uint256 value) internal {\\n uint256 newAllowance = token.allowance(address(this), spender).sub(value, \\\"SafeERC20: decreased allowance below zero\\\");\\n _callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender, newAllowance));\\n }\\n\\n /**\\n * @dev Imitates a Solidity high-level call (i.e. a regular function call to a contract), relaxing the requirement\\n * on the return value: the return value is optional (but if data is returned, it must not be false).\\n * @param token The token targeted by the call.\\n * @param data The call data (encoded using abi.encode or one of its variants).\\n */\\n function

_callOptionalReturn(IERC20 token, bytes memory data) private {\\n // We need to perform a low level call here, to bypass Solidity's return data size checking mechanism, since\\n // we're implementing it ourselves. We use {Address.functionCall} to perform this call, which verifies that\\n // the target address contains contract code and also asserts for success in the low-level call.\\n\\n bytes memory returndata = address(token).functionCall(data, \\\"SafeERC20: low-level call failed\\\");\\n if (returndata.length > 0) { // Return data is optional\\n // solhint-disable-next-line max-line-length\\n require(abi.decode(returndata, (bool)), \\\"SafeERC20: ERC20 operation did not succeed\\\");\\n }\\n }\\n}\\n\",\"keccak256\":\"0xf12dfbe97e6276980b83d2830bb0eb75e0cf4f3e626c2471137f82158ae6a0fc\",\"license\":\"MIT\"},\"@openzeppelin/contracts/utils/Address.sol\":{\"content\":\"// SPDX-License-Identifier: MIT\\n\\npragma solidity >=0.6.2 <0.8.0;\\n\\n/**\\n * @dev Collection of functions related to the address type\\n */\\nlibrary Address {\\n /**\\n * @dev Returns true if `account` is a contract.\\n *\\n * [IMPORTANT]\\n * ====\\n * It is unsafe to assume that an address for which this function returns\\n * false is an externally-owned account (EOA) and not a contract.\\n *\\n * Among others, `isContract` will return false for the following\\n * types of addresses:\\n *\\n * - an externally-owned account\\n * - a contract in construction\\n * - an address where a contract will be created\\n * - an address where a contract lived, but was destroyed\\n * ====\\n */\\n function isContract(address account) internal view returns (bool) {\\n // This method relies on extcodesize, which returns 0 for contracts in\\n // construction, since the code is only stored at the end of the\\n // constructor execution.\\n\\n uint256 size;\\n // solhint-disable-next-line no-inline-assembly\\n assembly { size := extcodesize(account) }\\n return size > 0;\\n }\\n\\n /**\\n * @dev Replacement for Solidity's `transfer`: sends `amount` wei to\\n * `recipient`, forwarding all available gas and reverting on errors.\\n *\\n * https://eips.ethereum.org/EIPS/eip-1884[EIP1884] increases the gas cost\\n * of certain opcodes, possibly making contracts go over the 2300 gas limit\\n * imposed by `transfer`, making them unable to receive funds via\\n * `transfer`. {sendValue} removes this limitation.\\n *\\n * https://diligence.consensys.net/posts/2019/09/stop-using-soliditys-transfer-now/[Learn more].\\n *\\n * IMPORTANT: because control is transferred to `recipient`, care must be\\n * taken to not create reentrancy vulnerabilities. Consider using\\n * {ReentrancyGuard} or the\\n * https://solidity.readthedocs.io/en/v0.5.11/security-considerations.html#use-the-checks-effects-interactions-pattern[checks-effects-interactions pattern].\\n */\\n function sendValue(address payable recipient, uint256 amount) internal {\\n require(address(this).balance >= amount, \\\"Address: insufficient balance\\\");\\n\\n // solhint-disable-next-line avoid-low-level-calls, avoid-call-value\\n (bool success,) = recipient.call{ value: amount }(\\\"\\\");\\n require(success, \\\"Address: unable to send value, recipient may have reverted\\\");\\n }\\n\\n /**\\n * @dev Performs a Solidity function call using a low level `call`. A\\n * plain`call` is an unsafe replacement for a function call: use this\\n * function instead.\\n *\\n * If `target` reverts with a revert reason, it is bubbled up by this\\n * function (like regular Solidity function calls).\\n *\\n * Returns the raw returned data. To convert to the

expected return value,\\n * use https://solidity.readthedocs.io/en/latest/units-and-global-variables.html?highlight=abi.decode#abi-encoding-and-decoding-functions[`abi.decode`].\\n *\\n * Requirements:\\n *\\n * - `target` must be a contract.\\n * - calling `target` with `data` must not revert.\\n *\\n * _Available since v3.1._\\n */\\n function functionCall(address target, bytes memory data) internal returns (bytes memory) {\\n return functionCall(target, data, \\\"Address: low-level call failed\\\");\\n }\\n\\n /**\\n * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`], but with\\n * `errorMessage` as a fallback revert reason when `target` reverts.\\n *\\n * _Available since v3.1._\\n */\\n function functionCall(address target, bytes memory data, string memory errorMessage) internal returns (bytes memory) {\\n return functionCallWithValue(target, data, 0, errorMessage);\\n }\\n\\n /**\\n * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],\\n * but also transferring `value` wei to `target`.\\n *\\n * Requirements:\\n *\\n * - the calling contract must have an ETH balance of at least `value`.\\n * - the called Solidity function must be `payable`.\\n *\\n * _Available since v3.1._\\n */\\n function functionCallWithValue(address target, bytes memory data, uint256 value) internal returns (bytes memory) {\\n return functionCallWithValue(target, data, value, \\\"Address: low-level call with value failed\\\");\\n }\\n\\n /**\\n * @dev Same as {xref-Address-functionCallWithValue-address-bytes-uint256-}[`functionCallWithValue`], but\\n * with `errorMessage` as a fallback revert reason when `target` reverts.\\n *\\n * _Available since v3.1._\\n */\\n function functionCallWithValue(address target, bytes memory data, uint256 value, string memory errorMessage) internal returns (bytes memory) {\\n require(address(this).balance >= value, \\\"Address: insufficient balance for call\\\");\\n require(isContract(target), \\\"Address: call to non-contract\\\");\\n\\n // solhint-disable-next-line avoid-low-level-calls\\n (bool success, bytes memory returndata) = target.call{ value: value }(data);\\n return _verifyCallResult(success, returndata, errorMessage);\\n }\\n\\n /**\\n * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],\\n * but performing a static call.\\n *\\n * _Available since v3.3._\\n */\\n function functionStaticCall(address target, bytes memory data) internal view returns (bytes memory) {\\n return functionStaticCall(target, data, \\\"Address: low-level static call failed\\\");\\n }\\n\\n /**\\n * @dev Same as {xref-Address-functionCall-address-bytes-string-}[`functionCall`],\\n * but performing a static call.\\n *\\n * _Available since v3.3._\\n */\\n function functionStaticCall(address target, bytes memory data, string memory errorMessage) internal view returns (bytes memory) {\\n require(isContract(target), \\\"Address: static call to non-contract\\\");\\n\\n // solhint-disable-next-line avoid-low-level-calls\\n (bool success, bytes memory returndata) = target.staticcall(data);\\n return _verifyCallResult(success, returndata, errorMessage);\\n }\\n\\n function _verifyCallResult(bool success, bytes memory returndata, string memory errorMessage) private pure returns(bytes memory) {\\n if (success) {\\n return returndata;\\n } else {\\n // Look for revert reason and bubble it up if present\\n if (returndata.length > 0) {\\n // The easiest way to bubble the revert reason is using memory via assembly\\n\\n // solhint-disable-next-line no-inline-assembly\\n assembly {\\n let returndata_size :=

mload(returndata)\\n revert(add(32, returndata), returndata_size)\\n }\\n } else {\\n revert(errorMessage);\\n }\\n }\\n }\\n}\\n\",\"keccak256\":\"0xa6a15ddddcbf29d2922a1e0d4151b5d2d33da24b93cc9ebc12390e0d855532f8\",\"license\":\"MIT\"},\"contracts/EasyAuction.sol\":{\"content\":\"pragma solidity >=0.6.8;\\r\\nimport \\\"@openzeppelin/contracts/token/ERC20/IERC20.sol\\\";\\r\\nimport \\\"@openzeppelin/contracts/token/ERC20/SafeERC20.sol\\\";\\r\\nimport \\\"./libraries/IterableOrderedOrderSet.sol\\\";\\r\\nimport \\\"@openzeppelin/contracts/math/Math.sol\\\";\\r\\nimport \\\"@openzeppelin/contracts/math/SafeMath.sol\\\";\\r\\nimport \\\"./libraries/IdToAddressBiMap.sol\\\";\\r\\nimport \\\"./libraries/SafeCast.sol\\\";\\r\\nimport \\\"@openzeppelin/contracts/access/Ownable.sol\\\";\\r\\nimport \\\"./interfaces/AllowListVerifier.sol\\\";\\r\\n\\r\\ncontract EasyAuction is Ownable {\\r\\n using SafeERC20 for IERC20;\\r\\n using SafeMath for uint64;\\r\\n using SafeMath for uint96;\\r\\n using SafeMath for uint256;\\r\\n using SafeCast for uint256;\\r\\n using IterableOrderedOrderSet for IterableOrderedOrderSet.Data;\\r\\n using IterableOrderedOrderSet for bytes32;\\r\\n using IdToAddressBiMap for IdToAddressBiMap.Data;\\r\\n\\r\\n modifier atStageOrderPlacement(uint256 auctionId) {\\r\\n require(\\r\\n block.timestamp < auctionData[auctionId].auctionEndDate,\\r\\n \\\"no longer in order placement phase\\\"\\r\\n);\\r\\n _;\\r\\n }\\r\\n\\r\\n modifier atStageOrderPlacementAndCancelation(uint256 auctionId) {\\r\\n require(\\r\\n block.timestamp < auctionData[auctionId].orderCancellationEndDate,\\r\\n \\\"no longer in order placement and cancelation phase\\\"\\r\\n);\\r\\n _;\\r\\n }\\r\\n\\r\\n modifier atStageSolutionSubmission(uint256 auctionId) {\\r\\n {\\r\\n uint256 auctionEndDate = auctionData[auctionId].auctionEndDate;\\r\\n require(\\r\\n auctionEndDate != 0 &&\\r\\n block.timestamp >= auctionEndDate &&\\r\\n auctionData[auctionId].clearingPriceOrder == bytes32(0),\\r\\n \\\"Auction not in solution submission phase\\\"\\r\\n);\\r\\n }\\r\\n _;\\r\\n }\\r\\n\\r\\n modifier atStageFinished(uint256 auctionId) {\\r\\n require(\\r\\n auctionData[auctionId].clearingPriceOrder != bytes32(0),\\r\\n \\\"Auction not yet finished\\\"\\r\\n);\\r\\n _;\\r\\n }\\r\\n\\r\\n event NewSellOrder(\\r\\n uint256 indexed auctionId,\\r\\n uint64 indexed userId,\\r\\n uint96 buyAmount,\\r\\n uint96 sellAmount\\r\\n);\\r\\n event CancellationSellOrder(\\r\\n uint256 indexed auctionId,\\r\\n uint64 indexed userId,\\r\\n uint96 buyAmount,\\r\\n uint96 sellAmount\\r\\n);\\r\\n event ClaimedFromOrder(\\r\\n uint256 indexed auctionId,\\r\\n uint64 indexed userId,\\r\\n uint96 buyAmount,\\r\\n uint96 sellAmount\\r\\n);\\r\\n event NewUser(uint64 indexed userId, address indexed userAddress);\\r\\n event NewAuction(\\r\\n uint256 indexed auctionId,\\r\\n IERC20 indexed _auctioningToken,\\r\\n IERC20 indexed _biddingToken,\\r\\n uint256 orderCancellationEndDate,\\r\\n uint256 auctionEndDate,\\r\\n uint64 userId,\\r\\n uint96 _auctionedSellAmount,\\r\\n uint96 _minBuyAmount,\\r\\n uint256 minimumBiddingAmountPerOrder,\\r\\n uint256 minFundingThreshold,\\r\\n address

allowListContract,\\r\\n bytes allowListData\\r\\n);\\r\\n event AuctionCleared(\\r\\n uint256 indexed auctionId,\\r\\n uint96 soldAuctioningTokens,\\r\\n uint96 soldBiddingTokens,\\r\\n bytes32 clearingPriceOrder\\r\\n);\\r\\n event UserRegistration(address indexed user, uint64 userId);\\r\\n\\r\\n struct AuctionData {\\r\\n IERC20 auctioningToken;\\r\\n IERC20 biddingToken;\\r\\n uint256 orderCancellationEndDate;\\r\\n uint256 auctionEndDate;\\r\\n bytes32 initialAuctionOrder;\\r\\n uint256 minimumBiddingAmountPerOrder;\\r\\n uint256 interimSumBidAmount;\\r\\n bytes32 interimOrder;\\r\\n bytes32 clearingPriceOrder;\\r\\n uint96 volumeClearingPriceOrder;\\r\\n bool minFundingThresholdNotReached;\\r\\n bool isAtomicClosureAllowed;\\r\\n uint256 feeNumerator;\\r\\n uint256 minFundingThreshold;\\r\\n }\\r\\n mapping(uint256 => IterableOrderedOrderSet.Data) internal sellOrders;\\r\\n mapping(uint256 => AuctionData) public auctionData;\\r\\n mapping(uint256 => address) public auctionAccessManager;\\r\\n mapping(uint256 => bytes) public auctionAccessData;\\r\\n\\r\\n IdToAddressBiMap.Data private registeredUsers;\\r\\n uint64 public numUsers;\\r\\n uint256 public auctionCounter;\\r\\n\\r\\n constructor() public Ownable() {}\\r\\n\\r\\n uint256 public feeNumerator = 0;\\r\\n uint256 public constant FEE_DENOMINATOR = 1000;\\r\\n uint64 public feeReceiverUserId = 1;\\r\\n\\r\\n function setFeeParameters(\\r\\n uint256 newFeeNumerator,\\r\\n address newfeeReceiverAddress\\r\\n) public onlyOwner() {\\r\\n require(\\r\\n newFeeNumerator <= 15,\\r\\n \\\"Fee is not allowed to be set higher than 1.5%\\\"\\r\\n);\\r\\n // caution: for currently running auctions, the feeReceiverUserId is changing as well.\\r\\n feeReceiverUserId = getUserId(newfeeReceiverAddress);\\r\\n feeNumerator = newFeeNumerator;\\r\\n }\\r\\n\\r\\n // @dev: function to intiate a new auction\\r\\n // Warning: In case the auction is expected to raise more than\\r\\n // 2^96 units of the biddingToken, don't start the auction, as\\r\\n // it will not be settlable. This corresponds to about 79\\r\\n // billion DAI.\\r\\n //\\r\\n // Prices between biddingToken and auctioningToken are expressed by a\\r\\n // fraction whose components are stored as uint96.\\r\\n function initiateAuction(\\r\\n IERC20 _auctioningToken,\\r\\n IERC20 _biddingToken,\\r\\n uint256 orderCancellationEndDate,\\r\\n uint256 auctionEndDate,\\r\\n uint96 _auctionedSellAmount,\\r\\n uint96 _minBuyAmount,\\r\\n uint256 minimumBiddingAmountPerOrder,\\r\\n uint256 minFundingThreshold,\\r\\n bool isAtomicClosureAllowed,\\r\\n address accessManagerContract,\\r\\n bytes memory accessManagerContractData\\r\\n) public returns (uint256) {\\r\\n // withdraws sellAmount + fees\\r\\n _auctioningToken.safeTransferFrom(\\r\\n msg.sender,\\r\\n address(this),\\r\\n _auctionedSellAmount.mul(FEE_DENOMINATOR.add(feeNumerator)).div(\\r\\n FEE_DENOMINATOR\\r\\n) //[0]\\r\\n);\\r\\n require(_auctionedSellAmount > 0, \\\"cannot auction zero tokens\\\");\\r\\n require(_minBuyAmount > 0, \\\"tokens cannot be auctioned for free\\\");\\r\\n require(\\r\\n minimumBiddingAmountPerOrder > 0,\\r\\n \\\"minimumBiddingAmountPerOrder is not allowed to be zero\\\"\\r\\n);\\r\\n require(\\r\\n orderCancellationEndDate <= auctionEndDate,\\r\\n \\\"time periods are not configured correctly\\\"\\r\\n);\\r\\n require(\\r\\n auctionEndDate > block.timestamp,\\r\\n \\\"auction end

date must be in the future\\\"\\r\\n);\\r\\n auctionCounter = auctionCounter.add(1);\\r\\n sellOrders[auctionCounter].initializeEmptyList();\\r\\n uint64 userId = getUserId(msg.sender);\\r\\n auctionData[auctionCounter] = AuctionData(\\r\\n _auctioningToken,\\r\\n _biddingToken,\\r\\n orderCancellationEndDate,\\r\\n auctionEndDate,\\r\\n IterableOrderedOrderSet.encodeOrder(\\r\\n userId,\\r\\n _minBuyAmount,\\r\\n _auctionedSellAmount\\r\\n),\\r\\n minimumBiddingAmountPerOrder,\\r\\n 0,\\r\\n IterableOrderedOrderSet.QUEUE_START,\\r\\n bytes32(0),\\r\\n 0,\\r\\n false,\\r\\n isAtomicClosureAllowed,\\r\\n feeNumerator,\\r\\n minFundingThreshold\\r\\n);\\r\\n auctionAccessManager[auctionCounter] = accessManagerContract;\\r\\n auctionAccessData[auctionCounter] = accessManagerContractData;\\r\\n emit NewAuction(\\r\\n auctionCounter,\\r\\n _auctioningToken,\\r\\n _biddingToken,\\r\\n orderCancellationEndDate,\\r\\n auctionEndDate,\\r\\n userId,\\r\\n _auctionedSellAmount,\\r\\n _minBuyAmount,\\r\\n minimumBiddingAmountPerOrder,\\r\\n minFundingThreshold,\\r\\n accessManagerContract,\\r\\n accessManagerContractData\\r\\n);\\r\\n return auctionCounter;\\r\\n }\\r\\n\\r\\n function placeSellOrders(\\r\\n uint256 auctionId,\\r\\n uint96[] memory _minBuyAmounts,\\r\\n uint96[] memory _sellAmounts,\\r\\n bytes32[] memory _prevSellOrders,\\r\\n bytes calldata allowListCallData\\r\\n) external atStageOrderPlacement(auctionId) returns (uint64 userId) {\\r\\n return\\r\\n _placeSellOrders(\\r\\n auctionId,\\r\\n _minBuyAmounts,\\r\\n _sellAmounts,\\r\\n _prevSellOrders,\\r\\n allowListCallData,\\r\\n msg.sender\\r\\n);\\r\\n }\\r\\n\\r\\n function placeSellOrdersOnBehalf(\\r\\n uint256 auctionId,\\r\\n uint96[] memory _minBuyAmounts,\\r\\n uint96[] memory _sellAmounts,\\r\\n bytes32[] memory _prevSellOrders,\\r\\n bytes calldata allowListCallData,\\r\\n address orderSubmitter\\r\\n) external atStageOrderPlacement(auctionId) returns (uint64 userId) {\\r\\n return\\r\\n _placeSellOrders(\\r\\n auctionId,\\r\\n _minBuyAmounts,\\r\\n _sellAmounts,\\r\\n _prevSellOrders,\\r\\n allowListCallData,\\r\\n orderSubmitter\\r\\n);\\r\\n }\\r\\n\\r\\n function _placeSellOrders(\\r\\n uint256 auctionId,\\r\\n uint96[] memory _minBuyAmounts,\\r\\n uint96[] memory _sellAmounts,\\r\\n bytes32[] memory _prevSellOrders,\\r\\n bytes calldata allowListCallData,\\r\\n address orderSubmitter\\r\\n) internal returns (uint64 userId) {\\r\\n {\\r\\n address allowListManger = auctionAccessManager[auctionId];\\r\\n if (allowListManger != address(0)) {\\r\\n require(\\r\\n AllowListVerifier(allowListManger).isAllowed(\\r\\n orderSubmitter,\\r\\n auctionId,\\r\\n allowListCallData\\r\\n) == AllowListVerifierHelper.MAGICVALUE,\\r\\n \\\"user not allowed to place order\\\"\\r\\n);\\r\\n }\\r\\n }\\r\\n {\\r\\n (\\r\\n ,\\r\\n uint96 buyAmountOfInitialAuctionOrder,\\r\\n uint96 sellAmountOfInitialAuctionOrder\\r\\n) = auctionData[auctionId].initialAuctionOrder.decodeOrder();\\r\\n for (uint256 i = 0; i < _minBuyAmounts.length; i++) {\\r\\n require(\\r\\n _minBuyAmounts[i].mul(buyAmountOfInitialAuctionOrder) <\\r\\n

sellAmountOfInitialAuctionOrder.mul(_sellAmounts[i]),\\r\\n \\\"limit price not better than mimimal offer\\\"\\r\\n);\\r\\n }\\r\\n }\\r\\n uint256 sumOfSellAmounts = 0;\\r\\n userId = getUserId(orderSubmitter);\\r\\n uint256 minimumBiddingAmountPerOrder =\\r\\n auctionData[auctionId].minimumBiddingAmountPerOrder;\\r\\n for (uint256 i = 0; i < _minBuyAmounts.length; i++) {\\r\\n require(\\r\\n _minBuyAmounts[i] > 0,\\r\\n \\\"_minBuyAmounts must be greater than 0\\\"\\r\\n);\\r\\n // orders should have a minimum bid size in order to limit the gas\\r\\n // required to compute the final price of the auction.\\r\\n require(\\r\\n _sellAmounts[i] > minimumBiddingAmountPerOrder,\\r\\n \\\"order too small\\\"\\r\\n);\\r\\n if (\\r\\n sellOrders[auctionId].insert(\\r\\n IterableOrderedOrderSet.encodeOrder(\\r\\n userId,\\r\\n _minBuyAmounts[i],\\r\\n _sellAmounts[i]\\r\\n),\\r\\n _prevSellOrders[i]\\r\\n)\\r\\n) {\\r\\n sumOfSellAmounts = sumOfSellAmounts.add(_sellAmounts[i]);\\r\\n emit NewSellOrder(\\r\\n auctionId,\\r\\n userId,\\r\\n _minBuyAmounts[i],\\r\\n _sellAmounts[i]\\r\\n);\\r\\n }\\r\\n }\\r\\n auctionData[auctionId].biddingToken.safeTransferFrom(\\r\\n msg.sender,\\r\\n address(this),\\r\\n sumOfSellAmounts\\r\\n); //[1]\\r\\n }\\r\\n\\r\\n function cancelSellOrders(uint256 auctionId, bytes32[] memory _sellOrders)\\r\\n public\\r\\n atStageOrderPlacementAndCancelation(auctionId)\\r\\n {\\r\\n uint64 userId = getUserId(msg.sender);\\r\\n uint256 claimableAmount = 0;\\r\\n for (uint256 i = 0; i < _sellOrders.length; i++) {\\r\\n // Note: we keep the back pointer of the deleted element so that\\r\\n // it can be used as a reference point to insert a new node.\\r\\n bool success =\\r\\n sellOrders[auctionId].removeKeepHistory(_sellOrders[i]);\\r\\n if (success) {\\r\\n (\\r\\n uint64 userIdOfIter,\\r\\n uint96 buyAmountOfIter,\\r\\n uint96 sellAmountOfIter\\r\\n) = _sellOrders[i].decodeOrder();\\r\\n require(\\r\\n userIdOfIter == userId,\\r\\n \\\"Only the user can cancel his orders\\\"\\r\\n);\\r\\n claimableAmount = claimableAmount.add(sellAmountOfIter);\\r\\n emit CancellationSellOrder(\\r\\n auctionId,\\r\\n userId,\\r\\n buyAmountOfIter,\\r\\n sellAmountOfIter\\r\\n);\\r\\n }\\r\\n }\\r\\n auctionData[auctionId].biddingToken.safeTransfer(\\r\\n msg.sender,\\r\\n claimableAmount\\r\\n); //[2]\\r\\n }\\r\\n\\r\\n function precalculateSellAmountSum(\\r\\n uint256 auctionId,\\r\\n uint256 iterationSteps\\r\\n) public atStageSolutionSubmission(auctionId) {\\r\\n (, , uint96 auctioneerSellAmount) =\\r\\n auctionData[auctionId].initialAuctionOrder.decodeOrder();\\r\\n uint256 sumBidAmount = auctionData[auctionId].interimSumBidAmount;\\r\\n bytes32 iterOrder = auctionData[auctionId].interimOrder;\\r\\n\\r\\n\\r\\n for (uint256 i = 0; i < iterationSteps; i++) {\\r\\n iterOrder = sellOrders[auctionId].next(iterOrder);\\r\\n (, , uint96 sellAmountOfIter) = iterOrder.decodeOrder();\\r\\n sumBidAmount = sumBidAmount.add(sellAmountOfIter);\\r\\n }\\r\\n\\r\\n require(\\r\\n iterOrder != IterableOrderedOrderSet.QUEUE_END,\\r\\n \\\"reached end of order list\\\"\\r\\n);\\r\\n\\r\\n // it is checked that not too many iteration steps were taken:\\r\\n // require that the sum of SellAmounts times the price of the last order\\r\\n // is not more than initially sold amount\\r\\n (, uint96 buyAmountOfIter, uint96 sellAmountOfIter) =\\r\\n

iterOrder.decodeOrder();\\r\\n require(\\r\\n sumBidAmount.mul(buyAmountOfIter) <\\r\\n auctioneerSellAmount.mul(sellAmountOfIter),\\r\\n \\\"too many orders summed up\\\"\\r\\n);\\r\\n\\r\\n auctionData[auctionId].interimSumBidAmount = sumBidAmount;\\r\\n auctionData[auctionId].interimOrder = iterOrder;\\r\\n }\\r\\n\\r\\n function settleAuctionAtomically(\\r\\n uint256 auctionId,\\r\\n uint96[] memory _minBuyAmount,\\r\\n uint96[] memory _sellAmount,\\r\\n bytes32[] memory _prevSellOrder,\\r\\n bytes calldata allowListCallData\\r\\n) public atStageSolutionSubmission(auctionId) {\\r\\n require(\\r\\n auctionData[auctionId].isAtomicClosureAllowed,\\r\\n \\\"not allowed to settle auction atomically\\\"\\r\\n);\\r\\n require(\\r\\n _minBuyAmount.length == 1 && _sellAmount.length == 1,\\r\\n \\\"Only one order can be placed atomically\\\"\\r\\n);\\r\\n uint64 userId = getUserId(msg.sender);\\r\\n require(\\r\\n auctionData[auctionId].interimOrder.smallerThan(\\r\\n IterableOrderedOrderSet.encodeOrder(\\r\\n userId,\\r\\n _minBuyAmount[0],\\r\\n _sellAmount[0]\\r\\n)\\r\\n),\\r\\n \\\"precalculateSellAmountSum is already too advanced\\\"\\r\\n);\\r\\n _placeSellOrders(\\r\\n auctionId,\\r\\n _minBuyAmount,\\r\\n _sellAmount,\\r\\n _prevSellOrder,\\r\\n allowListCallData,\\r\\n msg.sender\\r\\n);\\r\\n settleAuction(auctionId);\\r\\n }\\r\\n\\r\\n // @dev function settling the auction and calculating the price\\r\\n function settleAuction(uint256 auctionId)\\r\\n public\\r\\n atStageSolutionSubmission(auctionId)\\r\\n returns (bytes32 clearingOrder)\\r\\n {\\r\\n (\\r\\n uint64 auctioneerId,\\r\\n uint96 minAuctionedBuyAmount,\\r\\n uint96 fullAuctionedAmount\\r\\n) = auctionData[auctionId].initialAuctionOrder.decodeOrder();\\r\\n\\r\\n uint256 currentBidSum = auctionData[auctionId].interimSumBidAmount;\\r\\n bytes32 currentOrder = auctionData[auctionId].interimOrder;\\r\\n uint256 buyAmountOfIter;\\r\\n uint256 sellAmountOfIter;\\r\\n uint96 fillVolumeOfAuctioneerOrder = fullAuctionedAmount;\\r\\n // Sum order up, until fullAuctionedAmount is fully bought or queue end is reached\\r\\n do {\\r\\n bytes32 nextOrder = sellOrders[auctionId].next(currentOrder);\\r\\n if (nextOrder == IterableOrderedOrderSet.QUEUE_END) {\\r\\n break;\\r\\n }\\r\\n currentOrder = nextOrder;\\r\\n (, buyAmountOfIter, sellAmountOfIter) = currentOrder.decodeOrder();\\r\\n currentBidSum = currentBidSum.add(sellAmountOfIter);\\r\\n } while (\\r\\n currentBidSum.mul(buyAmountOfIter) <\\r\\n fullAuctionedAmount.mul(sellAmountOfIter)\\r\\n);\\r\\n\\r\\n if (\\r\\n currentBidSum > 0 &&\\r\\n currentBidSum.mul(buyAmountOfIter) >=\\r\\n fullAuctionedAmount.mul(sellAmountOfIter)\\r\\n) {\\r\\n // All considered/summed orders are sufficient to close the auction fully\\r\\n // at price between current and previous orders.\\r\\n uint256 uncoveredBids =\\r\\n currentBidSum.sub(\\r\\n fullAuctionedAmount.mul(sellAmountOfIter).div(\\r\\n buyAmountOfIter\\r\\n)\\r\\n);\\r\\n\\r\\n if (sellAmountOfIter >= uncoveredBids) {\\r\\n //[13]\\r\\n // Auction fully filled via partial match of currentOrder\\r\\n uint256 sellAmountClearingOrder =\\r\\n sellAmountOfIter.sub(uncoveredBids);\\r\\n auctionData[auctionId]\\r\\n

.volumeClearingPriceOrder = sellAmountClearingOrder\\r\\n .toUint96();\\r\\n currentBidSum = currentBidSum.sub(uncoveredBids);\\r\\n clearingOrder = currentOrder;\\r\\n } else {\\r\\n //[14]\\r\\n // Auction fully filled via price strictly between currentOrder and the order\\r\\n // immediately before. For a proof, see the security-considerations.md\\r\\n currentBidSum = currentBidSum.sub(sellAmountOfIter);\\r\\n clearingOrder = IterableOrderedOrderSet.encodeOrder(\\r\\n 0,\\r\\n fullAuctionedAmount,\\r\\n currentBidSum.toUint96()\\r\\n);\\r\\n }\\r\\n } else {\\r\\n // All considered/summed orders are not sufficient to close the auction fully at price of last order //[18]\\r\\n // Either a higher price must be used or auction is only partially filled\\r\\n\\r\\n if (currentBidSum > minAuctionedBuyAmount) {\\r\\n //[15]\\r\\n // Price higher than last order would fill the auction\\r\\n clearingOrder = IterableOrderedOrderSet.encodeOrder(\\r\\n 0,\\r\\n fullAuctionedAmount,\\r\\n currentBidSum.toUint96()\\r\\n);\\r\\n } else {\\r\\n //[16]\\r\\n // Even at the initial auction price, the auction is partially filled\\r\\n clearingOrder = IterableOrderedOrderSet.encodeOrder(\\r\\n 0,\\r\\n fullAuctionedAmount,\\r\\n minAuctionedBuyAmount\\r\\n);\\r\\n fillVolumeOfAuctioneerOrder = currentBidSum\\r\\n .mul(fullAuctionedAmount)\\r\\n .div(minAuctionedBuyAmount)\\r\\n .toUint96();\\r\\n }\\r\\n }\\r\\n auctionData[auctionId].clearingPriceOrder = clearingOrder;\\r\\n\\r\\n if (auctionData[auctionId].minFundingThreshold > currentBidSum) {\\r\\n auctionData[auctionId].minFundingThresholdNotReached = true;\\r\\n }\\r\\n processFeesAndAuctioneerFunds(\\r\\n auctionId,\\r\\n fillVolumeOfAuctioneerOrder,\\r\\n auctioneerId,\\r\\n fullAuctionedAmount\\r\\n);\\r\\n emit AuctionCleared(\\r\\n auctionId,\\r\\n fillVolumeOfAuctioneerOrder,\\r\\n uint96(currentBidSum),\\r\\n clearingOrder\\r\\n);\\r\\n // Gas refunds\\r\\n auctionAccessManager[auctionId] = address(0);\\r\\n delete auctionAccessData[auctionId];\\r\\n auctionData[auctionId].initialAuctionOrder = bytes32(0);\\r\\n auctionData[auctionId].interimOrder = bytes32(0);\\r\\n auctionData[auctionId].interimSumBidAmount = uint256(0);\\r\\n auctionData[auctionId].minimumBiddingAmountPerOrder = uint256(0);\\r\\n }\\r\\n\\r\\n function claimFromParticipantOrder(\\r\\n uint256 auctionId,\\r\\n bytes32[] memory orders\\r\\n)\\r\\n public\\r\\n atStageFinished(auctionId)\\r\\n returns (\\r\\n uint256 sumAuctioningTokenAmount,\\r\\n uint256 sumBiddingTokenAmount\\r\\n)\\r\\n {\\r\\n for (uint256 i = 0; i < orders.length; i++) {\\r\\n // Note: we don't need to keep any information about the node since\\r\\n // no new elements need to be inserted.\\r\\n require(\\r\\n sellOrders[auctionId].remove(orders[i]),\\r\\n \\"order is no longer claimable\\"\\r\\n);\\r\\n }\\r\\n AuctionData memory auction = auctionData[auctionId];\\r\\n (, uint96 priceNumerator, uint96 priceDenominator) =\\r\\n auction.clearingPriceOrder.decodeOrder();\\r\\n (uint64 userId, ,) = orders[0].decodeOrder();\\r\\n bool minFundingThresholdNotReached =\\r\\n auctionData[auctionId].minFundingThresholdNotReached;\\r\\n for (uint256 i = 0; i < orders.length; i++) {\\r\\n (uint64 userIdOrder, uint96 buyAmount, uint96 sellAmount) =\\r\\n orders[i].decodeOrder();\\r\\n require(\\r\\n userIdOrder == userId,\\r\\n \\"only allowed to claim

for same user\\\"\\r\\n);\\r\\n if (minFundingThresholdNotReached) {\\r\\n //[10]\\r\\n sumBiddingTokenAmount = sumBiddingTokenAmount.add(sellAmount);\\r\\n } else {\\r\\n //[23]\\r\\n if (orders[i] == auction.clearingPriceOrder) {\\r\\n //[25]\\r\\n sumAuctioningTokenAmount = sumAuctioningTokenAmount.add(\\r\\n auction\\r\\n .volumeClearingPriceOrder\\r\\n .mul(priceNumerator)\\r\\n .div(priceDenominator)\\r\\n);\\r\\n sumBiddingTokenAmount = sumBiddingTokenAmount.add(\\r\\n sellAmount.sub(auction.volumeClearingPriceOrder)\\r\\n);\\r\\n } else {\\r\\n if (orders[i].smallerThan(auction.clearingPriceOrder)) {\\r\\n //[17]\\r\\n sumAuctioningTokenAmount = sumAuctioningTokenAmount.add(\\r\\n sellAmount.mul(priceNumerator).div(priceDenominator)\\r\\n);\\r\\n } else {\\r\\n //[24]\\r\\n sumBiddingTokenAmount = sumBiddingTokenAmount.add(\\r\\n sellAmount\\r\\n);\\r\\n }\\r\\n }\\r\\n }\\r\\n emit ClaimedFromOrder(auctionId, userId, buyAmount, sellAmount);\\r\\n }\\r\\n sendOutTokens(\\r\\n auctionId,\\r\\n sumAuctioningTokenAmount,\\r\\n sumBiddingTokenAmount,\\r\\n userId\\r\\n); //[3]\\r\\n }\\r\\n\\r\\n function processFeesAndAuctioneerFunds(\\r\\n uint256 auctionId,\\r\\n uint256 fillVolumeOfAuctioneerOrder,\\r\\n uint64 auctioneerId,\\r\\n uint96 fullAuctionedAmount\\r\\n) internal {\\r\\n uint256 feeAmount =\\r\\n fullAuctionedAmount.mul(auctionData[auctionId].feeNumerator).div(\\r\\n FEE_DENOMINATOR\\r\\n); //[20]\\r\\n if (auctionData[auctionId].minFundingThresholdNotReached) {\\r\\n sendOutTokens(\\r\\n auctionId,\\r\\n fullAuctionedAmount.add(feeAmount),\\r\\n 0,\\r\\n auctioneerId\\r\\n); //[4]\\r\\n } else {\\r\\n //[11]\\r\\n (, uint96 priceNumerator, uint96 priceDenominator) =\\r\\n auctionData[auctionId].clearingPriceOrder.decodeOrder();\\r\\n uint256 unsettledAuctionTokens =\\r\\n fullAuctionedAmount.sub(fillVolumeOfAuctioneerOrder);\\r\\n uint256 auctioningTokenAmount =\\r\\n unsettledAuctionTokens.add(\\r\\n feeAmount.mul(unsettledAuctionTokens).div(\\r\\n fullAuctionedAmount\\r\\n)\\r\\n);\\r\\n uint256 biddingTokenAmount =\\r\\n fillVolumeOfAuctioneerOrder.mul(priceDenominator).div(\\r\\n priceNumerator\\r\\n);\\r\\n sendOutTokens(\\r\\n auctionId,\\r\\n auctioningTokenAmount,\\r\\n biddingTokenAmount,\\r\\n auctioneerId\\r\\n); //[5]\\r\\n sendOutTokens(\\r\\n auctionId,\\r\\n feeAmount.mul(fillVolumeOfAuctioneerOrder).div(\\r\\n fullAuctionedAmount\\r\\n),\\r\\n 0,\\r\\n feeReceiverUserId\\r\\n); //[7]\\r\\n }\\r\\n }\\r\\n\\r\\n function sendOutTokens(\\r\\n uint256 auctionId,\\r\\n uint256 auctioningTokenAmount,\\r\\n uint256 biddingTokenAmount,\\r\\n uint64 userId\\r\\n) internal {\\r\\n address userAddress = registeredUsers.getAddressAt(userId);\\r\\n if (auctioningTokenAmount > 0) {\\r\\n auctionData[auctionId].auctioningToken.safeTransfer(\\r\\n userAddress,\\r\\n auctioningTokenAmount\\r\\n);\\r\\n }\\r\\n if (biddingTokenAmount > 0) {\\r\\n auctionData[auctionId].biddingToken.safeTransfer(\\r\\n userAddress,\\r\\n biddingTokenAmount\\r\\n);\\r\\n }\\r\\n }\\r\\n\\r\\n function registerUser(address user) public

returns (uint64 userId) {\\r\\n numUsers = numUsers.add(1).toUint64();\\r\\n require(\\r\\n registeredUsers.insert(numUsers, user),\\r\\n \\\"User already registered\\\"\\r\\n);\\r\\n userId = numUsers;\\r\\n emit UserRegistration(user, userId);\\r\\n }\\r\\n\\r\\n function getUserId(address user) public returns (uint64 userId) {\\r\\n if (registeredUsers.hasAddress(user)) {\\r\\n userId = registeredUsers.getId(user);\\r\\n } else {\\r\\n userId = registerUser(user);\\r\\n emit NewUser(userId, user);\\r\\n }\\r\\n }\\r\\n\\r\\n function getSecondsRemainingInBatch(uint256 auctionId)\\r\\n public\\r\\n view\\r\\n returns (uint256)\\r\\n {\\r\\n if (auctionData[auctionId].auctionEndDate < block.timestamp) {\\r\\n return 0;\\r\\n }\\r\\n return auctionData[auctionId].auctionEndDate.sub(block.timestamp);\\r\\n }\\r\\n\\r\\n function containsOrder(uint256 auctionId, bytes32 order)\\r\\n public\\r\\n view\\r\\n returns (bool)\\r\\n {\\r\\n return sellOrders[auctionId].contains(order);\\r\\n }\\r\\n}\\r\\n\",\"keccak256\":\"0x8c14497bd253bed46a26f4858fffc60dbc7705475cd3ba2b952a40d7c0ad726c\"},\"contracts/interfaces/AllowListVerifier.sol\":{\"content\":\"// SPDX-License-Identifier: LGPL-3.0-or-later\\r\\npragma solidity >=0.6.8;\\r\\n\\r\\nlibrary AllowListVerifierHelper {\\r\\n /// @dev Value returned by a call to `isAllowed` if the check\\r\\n /// was successful. The value is defined as:\\r\\n /// bytes4(keccak256(\\\"isAllowed(address,uint256,bytes)\\\"))\\r\\n bytes4 internal constant MAGICVALUE = 0x19a05a7e;\\r\\n}\\r\\n\\r\\n///\\r\\n/// @dev Standardized interface for an allowList manager for easyAuction\\r\\n/// The interface was inspired by EIP-1271\\r\\ninterface AllowListVerifier {\\r\\n /// @dev Should return whether the a specific user has access to an auction\\r\\n /// by returning the magic value from AllowListVerifierHelper\\r\\n function isAllowed(\\r\\n address user,\\r\\n uint256 auctionId,\\r\\n bytes calldata callData\\r\\n) external view returns (bytes4);\\r\\n}\\r\\n\",\"keccak256\":\"0xe646b09ca1a8756d2b10c4d50370cecb6a88c4c63bf56a5f83af3794c7f41a80\",\"license\":\"LGPL-3.0-or-later\"},\"contracts/libraries/IdToAddressBiMap.sol\":{\"content\":\"pragma solidity ^0.6.0;\\r\\n\\r\\n//\\r\\n// Contract does not have test coverage, as it was nearly copied from:\\r\\n// https://github.com/gnosis/solidity-data-structures/blob/master/contracts/libraries/IdToAddressBiMap.sol\\r\\n// The only change is uint16 -> uint64\\r\\n//\\r\\n\\r\\nlibrary IdToAddressBiMap {\\r\\n struct Data {\\r\\n mapping(uint64 => address) idToAddress;\\r\\n mapping(address => uint64) addressToId;\\r\\n }\\r\\n\\r\\n function hasId(Data storage self, uint64 id) internal view returns (bool) {\\r\\n return self.idToAddress[id + 1] != address(0);\\r\\n }\\r\\n\\r\\n function hasAddress(Data storage self, address addr)\\r\\n internal\\r\\n view\\r\\n returns (bool)\\r\\n {\\r\\n return self.addressToId[addr] != 0;\\r\\n }\\r\\n\\r\\n function getAddressAt(Data storage self, uint64 id)\\r\\n internal\\r\\n view\\r\\n returns (address)\\r\\n {\\r\\n require(hasId(self, id), \\\"Must have ID to get Address\\\");\\r\\n return self.idToAddress[id + 1];\\r\\n }\\r\\n\\r\\n function getId(Data storage self, address addr)\\r\\n internal\\r\\n view\\r\\n

returns (uint64)\\r\\n {\\r\\n require(hasAddress(self, addr), \\\"Must have Address to get ID\\\");\\r\\n return self.addressToId[addr] - 1;\\r\\n }\\r\\n\\r\\n function insert(\\r\\n Data storage self,\\r\\n uint64 id,\\r\\n address addr\\r\\n) internal returns (bool) {\\r\\n require(addr != address(0), \\\"Cannot insert zero address\\\");\\r\\n require(id != uint64(-1), \\\"Cannot insert max uint64\\\");\\r\\n // Ensure bijectivity of the mappings\\r\\n if (\\r\\n self.addressToId[addr] != 0 ||\\r\\n self.idToAddress[id + 1] != address(0)\\r\\n) {\\r\\n return false;\\r\\n }\\r\\n self.idToAddress[id + 1] = addr;\\r\\n self.addressToId[addr] = id + 1;\\r\\n return true;\\r\\n }\\r\\n}\\r\\n\",\"keccak256\":\"0x44c23c11087ac487704504e99658ae1303d1cad5d5ba0b171a61adf04e2cd5ea\"},\"contracts/libraries/IterableOrderedOrderSet.sol\":{\"content\":\"pragma solidity >=0.6.8;\\r\\n\\r\\nimport \\\"@openzeppelin/contracts/math/SafeMath.sol\\\";\\r\\n\\r\\nlibrary IterableOrderedOrderSet {\\r\\n using SafeMath for uint96;\\r\\n using IterableOrderedOrderSet for bytes32;\\r\\n\\r\\n // represents smallest possible value for an order under comparison of fn smallerThan()\\r\\n bytes32 internal constant QUEUE_START =\\r\\n 0x0000000000000000000000000000000000000000000000000000000000000001;\\r\\n // represents highest possible value for an order under comparison of fn smallerThan()\\r\\n bytes32 internal constant QUEUE_END =\\r\\n 0xffffffffffffffffffffffffffffffffffffffff000000000000000000000001;\\r\\n\\r\\n /// The struct is used to implement a modified version of a doubly linked\\r\\n /// list with sorted elements. The list starts from QUEUE_START to\\r\\n /// QUEUE_END, and each node keeps track of its predecessor and successor.\\r\\n /// Nodes can be added or removed.\\r\\n ///\\r\\n /// `next` and `prev` have a different role. The list is supposed to be\\r\\n /// traversed with `next`. If `next` is empty, the node is not part of the\\r\\n /// list. However, `prev` might be set for elements that are not in the\\r\\n /// list, which is why it should not be used for traversing. Having a `prev`\\r\\n /// set for elements not in the list is used to keep track of the history of\\r\\n /// the position in the list of a removed element.\\r\\n struct Data {\\r\\n mapping(bytes32 => bytes32) nextMap;\\r\\n mapping(bytes32 => bytes32) prevMap;\\r\\n }\\r\\n\\r\\n struct Order {\\r\\n uint64 owner;\\r\\n uint96 buyAmount;\\r\\n uint96 sellAmount;\\r\\n }\\r\\n\\r\\n function initializeEmptyList(Data storage self) internal {\\r\\n self.nextMap[QUEUE_START] = QUEUE_END;\\r\\n self.prevMap[QUEUE_END] = QUEUE_START;\\r\\n }\\r\\n\\r\\n function isEmpty(Data storage self) internal view returns (bool) {\\r\\n return self.nextMap[QUEUE_START] == QUEUE_END;\\r\\n }\\r\\n\\r\\n function insert(\\r\\n Data storage self,\\r\\n bytes32 elementToInsert,\\r\\n bytes32 elementBeforeNewOne\\r\\n) internal returns (bool) {\\r\\n (, , uint96 denominator) = decodeOrder(elementToInsert);\\r\\n require(denominator != uint96(0), \\\"Inserting zero is not supported\\\");\\r\\n require(\\r\\n elementToInsert != QUEUE_START && elementToInsert != QUEUE_END,\\r\\n \\\"Inserting element is not valid\\\"\\r\\n);\\r\\n if (contains(self, elementToInsert)) {\\r\\n return false;\\r\\n }\\r\\n if (\\r\\n elementBeforeNewOne != QUEUE_START &&\\r\\n self.prevMap[elementBeforeNewOne] == bytes32(0)\\r\\n) {\\r\\n return false;\\r\\n }\\r\\n if (!elementBeforeNewOne.smallerThan(elementToInsert)) {\\r\\n return

false;\\r\\n }\\r\\n\\r\\n // `elementBeforeNewOne` might have been removed during the time it\\r\\n // took to the transaction calling this function to be mined, so\\r\\n // the new order cannot be appended directly to this. We follow the\\r\\n // history of previous links backwards until we find an element in\\r\\n // the list from which to start our search.\\r\\n // Note that following the link backwards returns elements that are\\r\\n // before `elementBeforeNewOne` in sorted order.\\r\\n while (self.nextMap[elementBeforeNewOne] == bytes32(0)) {\\r\\n elementBeforeNewOne = self.prevMap[elementBeforeNewOne];\\r\\n }\\r\\n\\r\\n // `elementBeforeNewOne` belongs now to the linked list. We search the\\r\\n // largest entry that is smaller than the element to insert.\\r\\n bytes32 previous;\\r\\n bytes32 current = elementBeforeNewOne;\\r\\n do {\\r\\n previous = current;\\r\\n current = self.nextMap[current];\\r\\n } while (current.smallerThan(elementToInsert));\\r\\n // Note: previous < elementToInsert < current\\r\\n self.nextMap[previous] = elementToInsert;\\r\\n self.prevMap[current] = elementToInsert;\\r\\n self.prevMap[elementToInsert] = previous;\\r\\n self.nextMap[elementToInsert] = current;\\r\\n\\r\\n return true;\\r\\n }\\r\\n\\r\\n /// The element is removed from the linked list, but the node retains\\r\\n /// information on which predecessor it had, so that a node in the chain\\r\\n /// can be reached by following the predecessor chain of deleted elements.\\r\\n function removeKeepHistory(Data storage self, bytes32 elementToRemove)\\r\\n internal\\r\\n returns (bool)\\r\\n {\\r\\n if (!contains(self, elementToRemove)) {\\r\\n return false;\\r\\n }\\r\\n bytes32 previousElement = self.prevMap[elementToRemove];\\r\\n bytes32 nextElement = self.nextMap[elementToRemove];\\r\\n self.nextMap[previousElement] = nextElement;\\r\\n self.prevMap[nextElement] = previousElement;\\r\\n self.nextMap[elementToRemove] = bytes32(0);\\r\\n return true;\\r\\n }\\r\\n\\r\\n /// Remove an element from the chain, clearing all related storage.\\r\\n /// Note that no elements should be inserted using as a reference point a\\r\\n /// node deleted after calling `remove`, since an element in the `prev`\\r\\n /// chain might be missing.\\r\\n function remove(Data storage self, bytes32 elementToRemove)\\r\\n internal\\r\\n returns (bool)\\r\\n {\\r\\n bool result = removeKeepHistory(self, elementToRemove);\\r\\n if (result) {\\r\\n self.prevMap[elementToRemove] = bytes32(0);\\r\\n }\\r\\n return result;\\r\\n }\\r\\n\\r\\n function contains(Data storage self, bytes32 value)\\r\\n internal\\r\\n view\\r\\n returns (bool)\\r\\n {\\r\\n if (value == QUEUE_START) {\\r\\n return false;\\r\\n }\\r\\n // Note: QUEUE_END is not contained in the list since it has no\\r\\n // successor.\\r\\n return self.nextMap[value] != bytes32(0);\\r\\n }\\r\\n\\r\\n // @dev orders are ordered by\\r\\n // 1. their price - buyAmount/sellAmount\\r\\n // 2. by the sellAmount\\r\\n // 3. their userId,\\r\\n function smallerThan(bytes32 orderLeft, bytes32 orderRight)\\r\\n internal\\r\\n pure\\r\\n returns (bool)\\r\\n {\\r\\n (\\r\\n uint64 userIdLeft,\\r\\n uint96 priceNumeratorLeft,\\r\\n uint96 priceDenominatorLeft\\r\\n) = decodeOrder(orderLeft);\\r\\n (\\r\\n uint64 userIdRight,\\r\\n uint96 priceNumeratorRight,\\r\\n uint96 priceDenominatorRight\\r\\n) = decodeOrder(orderRight);\\r\\n\\r\\n if (\\r\\n priceNumeratorLeft.mul(priceDenominatorRight) <\\r\\n priceNumeratorRight.mul(priceDenominatorLeft)\\r\\n) return true;\\r\\n if (\\r\\n

priceNumeratorLeft.mul(priceDenominatorRight) >\\r\\n
priceNumeratorRight.mul(priceDenominatorLeft)\\r\\n) return false;\\r\\n\\r\\n if
(priceNumeratorLeft < priceNumeratorRight) return true;\\r\\n if (priceNumeratorLeft >
priceNumeratorRight) return false;\\r\\n require(\\r\\n userIdLeft != userIdRight,\\r\\n \\\"user is
not allowed to place same order twice\\\"\\r\\n);\\r\\n if (userIdLeft < userIdRight) {\\r\\n return
true;\\r\\n }\\r\\n return false;\\r\\n }\\r\\n\\r\\n function first(Data storage self) internal view
returns (bytes32) {\\r\\n require(!isEmpty(self), \\\"Trying to get first from empty set\\\");\\r\\n
return self.nextMap[QUEUE_START];\\r\\n }\\r\\n\\r\\n function next(Data storage self, bytes32
value)\\r\\n internal\\r\\n view\\r\\n returns (bytes32)\\r\\n {\\r\\n require(value != QUEUE_END,
\\\"Trying to get next of last element\\\");\\r\\n bytes32 nextElement = self.nextMap[value];\\r\\n
require(\\r\\n nextElement != bytes32(0),\\r\\n \\\"Trying to get next of non-existent
element\\\"\\r\\n);\\r\\n return nextElement;\\r\\n }\\r\\n\\r\\n function decodeOrder(bytes32
_orderData)\\r\\n internal\\r\\n pure\\r\\n returns (\\r\\n uint64 userId,\\r\\n uint96
buyAmount,\\r\\n uint96 sellAmount\\r\\n)\\r\\n {\\r\\n // Note: converting to uint discards the
binary digits that do not fit\\r\\n // the type.\\r\\n userId = uint64(uint256(_orderData) >>
192);\\r\\n buyAmount = uint96(uint256(_orderData) >> 96);\\r\\n sellAmount =
uint96(uint256(_orderData));\\r\\n }\\r\\n\\r\\n function encodeOrder(\\r\\n uint64 userId,\\r\\n
uint96 buyAmount,\\r\\n uint96 sellAmount\\r\\n) internal pure returns (bytes32) {\\r\\n
return\\r\\n bytes32(\\r\\n (uint256(userId) << 192) +\\r\\n (uint256(buyAmount) << 96) +\\r\\n
uint256(sellAmount)\\r\\n);\\r\\n
}\\r\\n}\\r\\n\",\"keccak256\":\"0x9510258dca14a3327710d516a1b5c0e7147e20ac8f5a29a9804
8d97878ff1dac\"},\"contracts/libraries/SafeCast.sol\":{\"content\":\"// SPDX-License-Identifier:
MIT\\r\\n\\r\\npragma solidity >=0.6.0 <0.8.0;\\r\\n\\r\\n/**\\r\\n * @dev Wrappers over Solidity's
uintXX/intXX casting operators with added overflow\\r\\n * checks.\\r\\n *\\r\\n * Logic was
copied and modified from here: https://github.com/OpenZeppelin/openzeppelin-
contracts/blob/master/contracts/utils/SafeCast.sol\\r\\n */\\r\\nlibrary SafeCast {\\r\\n function
toUint96(uint256 value) internal pure returns (uint96) {\\r\\n require(value < 2**96, \\\"SafeCast:
value doesn't fit in 96 bits\\\");\\r\\n return uint96(value);\\r\\n }\\r\\n\\r\\n function
toUint64(uint256 value) internal pure returns (uint64) {\\r\\n require(value < 2**64, \\\"SafeCast:
value doesn't fit in 64 bits\\\");\\r\\n return uint64(value);\\r\\n
}\\r\\n}\\r\\n\",\"keccak256\":\"0xe9d43c6764bad8042246919c329bb1cf5385ba190374f64f1359
82b8d1c0bbdd\",\"license\":\"MIT\"}},\"version\":1}", "bytecode":
"0x6080604052600060095560016001600a60006101000a81548167ffffffffffffffff021916908367ffffffffff
ffffff160217905550348015620000405760008fd5b5060006200053620000f760201b60201c5
65b905080600080610100a81548173ffffffffffffffffffffffffffffffffffffffff021916908373ffffffffffffffffffffffffffffffffffffffff
ffffffffff1602179055508073ffffffffffffffffffffffffffffffffffffffff16600073ffffffffffffffffffffffffffffffffffffffff167f8be00
79c531659141344cd1fd0a4f28419497f9722a3daafe3b4186f6b6457e0604051604051809103
90a350620000ff565b600033905090565b615b6e806200010f6000396000f3fe6080604052348
01561001057600080fd5b506004361061015760003560e01c80637882deaf116100c357806

3d73792a91161007c578063d73792a914610ccf578063e4a59ef414610ced578063e86dea4a1
4610d45578063ec20d0bb14610d63578063f2fde38b14610fe0578063f59c2f06146110245761
0158565b80637882deaf146106425780637ed18b701461071f5780638da5cb5b146107e15780
6391cfc1d414610815578063a7e7664414610a54578063d225269c14610a7257610158565b8
0633e12905f116101155780633e12905f1461046157806340b20b09146104957806355fc62d
2146104e75780635cefb291146105c257806363c699a4146105ea578063715018a6146106385
7610158565b80630a4cd6c91461015d57806315d37b4b146102f157806319a50f49146103335
780632199d5cd1461035b5780632b956ff7146103bd5780632e9936111461041f575b600080fd
5b6102db60048036036101608110156101745760080fd5b810190808035737fffffffffffffffffffffffffffffff
fffffffff1690602001909291908035737fffffffffffffffffffffffffffffffffffffff16906020019092919080359060200190929190803590602
0019092919080359060200190929190803560bfffffffffffffffffffff16906020019092919080356bffffff
ffffffffffffffff16906020019092919080359060200190929190803590602001909291908035151590602001909291908035737fffffffffffffffffffffffffffffffffffffff16906020019092919080359060200190640100000000811115610255576008fd5b820183602082011115610267576008fd5b803590602001918460018302840111640100000000831117156102895760080fd5b91908080601f016020809104026020016040519081016040528093929190818152602001838380828437600081840152601f19601f8201169050808301925050505050505091929192905050506110cb565b6040518082815260200191505060405180910390f35b61031d60048036036020811015610307576008fd5b810190808035906020019092919050505061178a565b604051808281526020019150506040518091039f35b61033b6117e4565b604051808267ffffffffffffffff16815260200191505060405180910390f35b61039d60048036036020811015610371576008fd5b810190808035737fffffffffffffffffffffffffffffffffffffff16906020019092919050505061178fe565b604051808267ffffffffffffffff16815260200191505060405180910390f35b6103ff60048036036020811015610357576008fd5b810190808035737fffffffffffffffffffffffffffffffffffffff16906020019092919050505061197856 5b604051808267ffffffffffffffff16815260200191505060405180910390f35b61044b60048036036020811015610435576008fd5b810190808035906020019092919050505061a0d565b604051808281526020019150506040518091039f35b6104976004803603604081101561047576008fd5b810190808035906020019092919080359060200190929190505050611fb4565b005b6104e5600480360360408110156104af576008fd5b810190808035906020019092919080359060200190929190803573ffffffffffffffffffffffffffffffffffffffff16906020019092919050505061225d6565b005b6105136004803603604081101561052757600080fd5b810190808035906020019092919050505061243456 5b604051808f73ffffffffffffffffffffffffffffffffffffffff1681526020018e73ffffffffffffffffffffffffffffffffffffffff1681526020018d8152602001 8c81526020018b81526020018a8152602001898152602001888152602001878152602001866bffffffffffffffffffffffff16815260200185151581526020018415158152602001838152602001828152602019e5050505050505050505050505050604051809103 90f35b6105ca612512565b604051808267ffffffffffffffff16815260200191505060405180910390f35b610620600480360360408110156106057600080fd5b810190808035906020019092919080359060200190929190505050612525c565b604051808215158152602001915050604051809103 90f35b61064061255b565b005b610702600480360360408110156106857600080fd5b810190808035906020019092

919080359060200190640100000000811115610 67f57600080fd5b8201836020820111156106
9157600080fd5b80359060200191846020830284011164010000000083111715 6106b3576000
80fd5b919080806020026020016040519081016040528093929190818152602001838360200
280828437600081840152601f19601f8201169050808301925050505050505050919291929050 5
0506126e1565b604051808381526020018281526020019250505060405180910390f35b6107
df60048036036040811015610735 57600080fd5b810190808035906020019092919080359060
200190640100000000811115610 75c57600080fd5b8201836020820111156107 6e57600080fd
5b80359060200191846020830284011164010000000083111715610 7905 7600080fd5b919080
806020026020016040519081016040528093929190818152602001838360200280828437600
081840152601f19601f8201169050808301925050505050505050919291929050505061 2d2d565
b005b6107e9612f90565b604051808273ffffffffffffffffffffffffffffffffffffffff1681526020019150506040 51
80910390f35b610a52600480360360a081101561082b57600080fd5b8101908080359060200 1
909291908035906020019064010000000811115610 85257600080fd5b820183602082011115
610 86457600080fd5b80359060200191846020830284011164010000000083111171561088657
600080fd5b919080806020026020016040519081016040528093929190818152602001838 36
020028082843760008184015 2601f19601f8201169050808301925050505050505050919291929
080359060200190640100000000811115610 8e657600080fd5b8201836020820111156108f85
7600080fd5b80359060200191846020830284011164010000000083111171561091a57600080f
d5b919080806020026020016040519081016040528093929190818152602001838360200280
828437600081840152601f19601f8201169050808301925050505050505050919291929080359
060200190640100000000811115610 97a57600080fd5b8201836020820111156109 8c57600080
fd5b80359060200191846020830284011164010000000083111715610 9ae57600080fd5b9190
808060200260200160405190810160405280939291908181526020018383602002808284376
000818401526 01f19601f8201169050808301925050505050505050919291929080359060200 19
064010000000811115610a0e57600080fd5b8201836020820111156 10a2057600080fd5b803
590602001918460018302840111640100000000831117156 10a4257600080fd5b90919293919
29390505050612fb9565b005b610a5c613221565b604051808281526020019150506040518 0
910390f35b610caf600480360360a0811015610a8857600080fd5b810190808035906020019 0
929190803590602001906401000000008111156 10aaf57600080fd5b8201836020820111156 1
0ac157600080fd5b80359060200191846020830284011164010000000083111715610ae35760
0080fd5b919080806020026020016040519081016040528093929190818152602001838360 2
002808284376000818401526 01f19601f8201169050808301925050505050505050919291929 08
035906020019064010000000811115610b4357600080fd5b8201836020820111156 10b55576
00080fd5b80359060200191846020830284011164010000000083111715610b7757600080fd5
b919080806020026020016040519081016040528093929190818152602001838360200280 82
843760008184015 2601f19601f8201169050808301925050505050505050919291929080359060
200190640100000000811115610bd757600080fd5b8201836020820111156 10be957600080fd
5b80359060200191846020830284011164010000000083111715610c0b57600080fd5b919080
806020026020016040519081016040528093929190818152602001838360200280828437600

0818401526 01f19601f8201169050808301925050505050505091929192908035906020019064010000000008111156 10c6b57600080fd5b8201836020820111156 10c7d57600080fd5b8035906020019184600183028401116401000000008311171 5610c9f57600080fd5b909192939192939050505061322756 5b604051808267ffffffffffffffff16815260200191505060405180910390f35b610cd76132b4565b6040518082815260200191505060405180910390f35b610d1960048036036020811015610d0357600080fd5b810190808035906020019092919050505061 32ba565b604051808273ffffffffffffffffffffffffffffffffffffffff16815260200191505060405180910390f35b610d4d6132ed565b6040518082815260200191505060405180910390f35b610fc0600480360360c0811015610d7957600080fd5b810190808035906020019092919080359060200190640100000000081111561 0da057600080fd5b8201836020820111156 10db257600080fd5b8035906020019184602083028401116401000000008311171 5610dd457600080fd5b919080806020026020016040519081016040528093929190818152602001838360200280828437600081840152601f19601f8201169050808301925050505050509192919290803590602001906401000000008111156 10e3457600080fd5b8201836020820111156 10e4657600080fd5b8035906020019184602083028401116401000000008311171 5610e6857600080fd5b919080806020026020016040519081016040528093929190818152602001838360200280828437600081840152601f19601f8201169050808301925050505050509192919290803590602001906401000000008111156 10ec857600080fd5b8201836020820111156 10eda57600080fd5b8035906020019184602083028401116401000000008311171 5610efc57600080fd5b919080806020026020016040519081016040528093929190818152602001838360200280828437600081840152601f19601f8201169050808301925050505050509192919290803590602001906401000000008111156 10f5c57600080fd5b8201836020820111156 10f6e57600080fd5b8035906020019184600183028401116401000000008311171 5610f9057600080fd5b9091929391929390803573ffffffffffffffffffffffffffffffffffffffff16906020019092919050505061323565b604051808267ffffffffffffffff16815260200191505060405180910390f35b611022600480360360208110156 10ff657600080fd5b8101908080357 3ffffffffffffffffffffffffffffffffffffffffffffffff16906020019092919050505061338156 5b005b611050600480360360208110156 1103a57600080fd5b810190808035906020019092919050505061358c565b604051808060200182810382528381815181526020019150805190602001908083836000 5b8381101561109057808201518184015260208101905061107556 5b505050509050908101906 01f1680156110bd5780820380516001836020036101000a031916815260200191505b5092505050604051809103 90f35b600611144333061111d6103e861110f6110f26009546103e861363c90919063ffffffff16565b8e6bfffffffffffffffffffffff166136c490919063ffffffff16565b61374a90919063ffffffff16565b8f73ffffffffffffffffffffffffffffffff16613794909392919063ffffffff16565b6000886bffffffffffffffffffffffff16116111c8576040517f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526023815260200180615840602391396040019150506040518091039 0fd5b60008611611288576040517f08c379a000000000000000

0000000000000000000000000000000000000000815260040180806020018281038252603
6815260200180615 9a66036913960400191505060405180910390fd5b888a11156112e15760
40517f08c379a0000000000000000000000000000000000000000000000000000000081526
0040180806020018281038252602981526020018061 5ac46029913960400191505060405180
910390fd5b428911611339576040517f08c379a0000000000000000000000000000000000000
0000000000000000000081526004018080602001828103825260268152602001806 1592f602
6913960400191505060405180910390fd5b61134f600160085461363c90919063ffffffff16565b6
0088190555061137260016000600854815260200190815260200160002061385556 5b600061
137d33611978565b9050604051806101c001604052808e73ffffffffffffffffffffffffffffffffffffffff16815260
20018d73ffffffffffffffffffffffffffffffffffffffff1681526020018c81526020018b81526020016113da838b8d
6138da565b8152602001888152602001600081526020016001600160001b81526020016000801b8
15260200160006bffffffffffffffffffffffff16815260200160001515815260200186151581526020016 0
0954815260200187815250600260006008548152602001908152602001600020600082015 18
160000160006101000a81548173ffffffffffffffffffffffffffffffffffffffff02191690837 3ffffffffffffffffffffffffffffffffffffffff
fff160217905550602082015181600101600061 01000a81548173ffffffffffffffffffffffffffffffffffffffff02191
6908373ffffffffffffffffffffffffffffffffffffffff160217905550604082015181600201556060820151816003 0
1556080820151816004015560a0820151816005015560c0820151816006015560e082015181
600701556101008201518160080155610120820151816009016000 6101000a8154816bffffffffff
ffffffffffffff02191690836bffffffffffffffffffffffff160217905550610140820151816009016 00c6101000a8
1548160ff0219169083151502179055506101608201518160090160 0d6101000a81548160ff02
191690831515021790555061018082015181600a01556101a082015181600b0155905050836
0036000600854815260200190815260200160002060006101000a81548173ffffffffffffffffffffffffffffffffffffffff
ffffffff02191690837 3ffffffffffffffffffffffffffffffffffffffff160217905550826004600060085481526020019
0815260200160002090805190602001906116 2b9291906155dc565b508b73ffffffffffffffffffffffffffffffffffffffff
ffffffff168d73ffffffffffffffffffffffffffffffffffffffff166008547f728d0fed13687f1840de94f5ae640eae49b43
eda26b7ffe97e55e79a0de6e40c8e8e868f8f8f8f8e8e604051808a81526020018981526020018
867ffffffffffffff168152602001876bffffffffffffffffffffff168152602001866bffffffffffffffffffffff1681526020
0185815260200184815260200183 73ffffffffffffffffffffffffffffffffffffffff16815260200180602001828103
8252838181518152602001915080519060200190808383600 05b838110156117335780820 15
1818401526020810190506117185 65b505050509050908101906 01f16801561176057808203
8051600183602003610 1000a031916815260200191505b509a505050505050505050505050604
05180910390a46008549150509b9a505050505050505050505050565b60004260026000848152
6020019081526020016000206003015410156117b3576000905061 17df565b6117dc42600260
0085815260200190815260200160002060030154613 91b90919063ffffffff16565b90505b91905
0565b60076000905490 6101000a900467ffffffffffffffff1681565b600061183c61183760016007600
090549061 01000a900467ffffffffffffffff1667ffffffffffffffff1661363c90919063ffffffff16565b613965565
b600760006101000a81548167ffffffffffffffff021916908367ffffffffffffffff160217905550611890600 76
0009054906101000a900467ffffffffffffffff168360056139d09092919063ffffffff16565b61190257604
0517f08c379a0000000000000000000000000000000000000000000000000000000081526 0

040180806020018281038252601781526020018 07f5573657220616c7265616479207265676
973746572656400000000000000000081525060200191505060405180910390fd5b60076000
9054906101000a900467ffffffffffffffff1690508173ffffffffffffffffffffffffffffffffffffffff167f6838f67cb358c332
087b73dbe769a6c869a5f87225236c16dda5d0013a77074c82604051808267ffffffffffffffff168152
602001915050604051809103 90a2919050565b600061198e826005613cf690919063ffffffff1656
5b156119ae576119a7826005613d6390919063ffffffff16565b9050611a08565b6119b7826117fe
565b90508173ffffffffffffffffffffffffffffffffffffffff168167ffffffffffffffff167f969d438b19b6b5fb3c0d3c6f1686
7e519fcec4233799d6be29d1c00f2045ba2660405160405180910390a35b919050565b600081
600600260008381526020019081526020016000206003015490506000811415801561 1a3c5
750804210155b8015611a6057506000801b60026000848152602001908152602001600020 60
080154145b611ab5576040517f08c379a0000000000000000000000000000000000000000000
0000000000000081526004018080602001828103825260288152602001806 15a4660289139
604001915050604051809103 90fd5b506000806000611ada600260008881526020019081526
020016000206004015461 3e42565b92509250925060006002600088815260200190815260 20
016000206006015490506000600260008981526020019081526020016000206007015490506
000806000859050 5b6000611b4685600160008f8152602001908152602001600020613e6890
919063ffffffff16565b90507fffffffffffffffffffffffffffffffffffffffff0000000000000000000000000160001b81141
5611b795750611c02565b809450611b8585613e42565b909150816bffffffffffffffffffffffff169150806
bffffffffffffffffffffffff1690508094508195505050611bc5838761363c90919063ffffffff16565b9550506
11be982876bffffffffffffffffffffffff166136c490919063ffffffff16565b611bfc84876136c490919063ffffffff
16565b10611b1f575b60008511801561 1c445750611c2e82876bffffffffffffffffffffffff166136c490919
063ffffffff16565b611c4184876136c490919063ffffffff16565b10155b15611d4d576000611c90611
c8185611c73868b6bffffffffffffffffffffffff166136c490919063ffffffff16565b61374a90919063ffffffff165
65b8761391b90919063ffffffff16565b9050808310611d1b576000611cae828561391b90919063f
ffffffff16565b9050611cb981 3f66565b600260008f8152602001908152602001600020600901 6
0006101000a8154816bffffffffffffffffffffffff02191690836bffffffffffffffffffffffff160217905550611d10828
861391b90919063ffffffff16565b9650859b5050611d47565b611d2e838761391b90919063ffffffff
16565b9550611d44600088611d3f89613f66565b6138da565b9a505b50611dd9565b866bffffffffff
ffffffffffffff16851115611d7e57611d77600087611d7288613f66565b6138da565b9950611dd8565
b611d8a600087896138da565b9950611dd5611dd0886bffffffffffffffffffffffff16611dc2896bffffffffffffffff
ffffffff16896136c490919063ffffffff16565b61374a90919063ffffffff16565b613f66565b90505b5b89
600260008d8152602001908152602001600020600801819055508460026000 8d81526020019
081526020016000206 00b01541115611e42576001600260008d8152602001908152602001600
0020600901600c6101000a81548160ff0219169083151502179055505b611e5c8b826bffffffffffff
ffffffffff168a89613fd5565b8a7f4d160a2a345f2faeb9ac2e65272820b8ca5473b80aabef416bdf7
e07ee7f591082878d60405180846bffffffffffffffffffffff16815260200 1836bffffffffffffffffffffff16815260
200182815260200193505050506040518091039 0a26000600360008d8152602001908152602
00160002060006101000a81548173ffffffffffffffffffffffffffffffffffffffff02191690837 3ffffffffffffffffffffffffffffff
ffffff160217905550600460008c8152602001908152602001600020600 0611f32919061565c56

5b6000801b600260008d81526020019081526020016000206004018190555060000801b60026
0008d81526020019081526020016000206007018190555060006002600008d81526020019081
52602001600020600601819055506000600260008d81526020019081526020016000206005 0
1819055505050505050505050919050565b816000600260008381526020019081526020016
000206003015490506000811415801561fe15750804210155b801561200557506000801b6
00260008481526020019081526020016000206008015414 5b61205a576040517f08c379a000
0000000000000000000000000000000000000000000000000000000081526004018080602001 8
2810382526028815260200180615a466028913960400191505060405180910390fd5b506000
61207c600260008681526020019081526020016000206004015461 3e42565b9250505060006
00260008681526020019081526020016000206006015490506000600260008781526020019 0
81526020016000206007015490506000 5b858110156121285761 20e682600160008a8152602
00190815260200160002061 3e6890919063ffffffff16565b915060006120f383613e42565b9250
5050612118816bffffffffffffffffffffffff168561363c90919063ffffffff16565b935050808600101915050
6120b9565b507fffffffffffffffffffffffffffffffffffffffffffffffff0000000000000000000000160001b8114156121c25
76040517f08c379a0000000000000000000000000000000000000000000000000000000081
526004018080602001828103825260198152602001807f72656163686564206e64206f6620
6f72646572206c6973740000000000000081525060200191505060405180910390fd5b600080
6121ce83613e42565b9250925050612202816bffffffffffffffffffffffff16866bffffffffffffffffffffffff166136c49
0919063ffffffff16565b612223836bffffffffffffffffffffffff16866136c490919063ffffffff16565b106122965
76040517f08c379a0000000000000000000000000000000000000000000000000000000081
526004018080602001828103825260198152602001807f746f6f206d616e79206f72646572732
073756d6d656420757000000000000000815250602001915050604051809103902 0
60008a81526020019081526020016000206006018190555082600260008a815260200190815
26020016000206007018190555050505050505050565b6122de6141c7565b73ffffffffffffffffffff
ffffffffffffffff1660008054906101000a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff
ff161461239e576040517f08c379a0000000000000000000000000000000000000000000000000
0000000008152600401808060200182810382526020815260200180 7f4f776e61626c653a20
63616c6c6572206973206e6f7420746865206f776e657281525060200191505060405180910390
90fd5b600f8211156123f8576040517f08c379a0000000000000000000000000000000000000
0000000000000000000815260040180806020018281038252602d81526020018061590260 2d
91396040019150506040518091039 0fd5b6124018161197856 5b600a60006101000a8154816
7ffffffffffffffff021916908367ffffffffffffffff160217905550816009819055505050565b6002602052806
000526040600020600091509050806000016000905490610100 0a900473ffffffffffffffffffffffffffffffffffffffff
ffffff16908060010160009054906101000a900473ffffffffffffffffffffffffffffffffffffffff16908060020154908
060030154908060040154908060050154908060060154908060070154908060080154908060
090160009054906101000a90046bffffffffffffffffffffffff16908060090160 0c9054906101000a900460
ff16908060090160 0d9054906101000a900460ff169080600a01549080600b015490508e565b6
00a60009054906101000a900467ffffffffffffffff1681565b60006125538260016000868152602001
908152602001600020614 1cf90919063ffffffff16565b905092915050565b6125636141c7565b73

ffffffffffffffffffffffffffffffffffff16600080549061010000a900473ffffffffffffffffffffffffffffffffffff1673ffffffffffffffff
ffffffffffffffffffff161461262357604051 7f08c379a0000000000000000000000000000000000000000
0000000000000000000815260040180806020018281038252602081526020018 07f4f776e616
26c653a2063616c6c6572206973206e6f7420746865206f776e657281525060200191505060 4
05180910390fd5b600073ffffffffffffffffffffffffffffffffffff16600080549061010000a900473ffffffffffffffffffff
ffffffffffffffff1673ffffffffffffffffffffffffffffffffffff167f8be0079c531659141344cd1fd0a4f28419497f972
2a3daafe3b4186f6b6457e060405160405180910390a3600080600061010000a81548173ffffffffff
ffffffffffffffffffffffffff021916908373ffffffffffffffffffffffffffffffffffff160217905550565b60008083600080
1b60026000838152602001908152602001600020600801541415612774576040517f08c379a
0000000000000000000000000000000000000000000000000000000815260040180806020
018281038252601881526020018 07f41756374696f6e206e6f7420796574206696e69736865
64000000000000000081525060200191505060405180910390fd5b60005b845181101561283
7576127b8858281518110612 78f57fe5b602002602001015160016000898152602001908152 6
020016000206142 0b90919063ffffffff16565b61282a576040517f08c379a00000000000000
0000000000000000000000000000000000000000000815260040180806020018281038252601c8
152602001807f6f72646572206973206e6f206c6f6e67657220636c61696d61626c6500000000
8152506020019150506040518091039 0fd5b8080600101915050612777565b506128406156a
4565b60026000878152602001908152602001600020604051806101c0016040529081600082
0160009054906101000a900473ffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffff1673ffffffff
ffffffffffffffffffffffffffff168152602001600182016000905490 6101000a900473ffffffffffffffffffffffffffffffffffff
fffff1673ffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffff168152602001600282015481526
020016003820154815260200160048201548152602001600582015481526020016006820154
8152602001600782015481526020016008820154815260200160098201600090549061010 0
a90046bffffffffffffffffffffff166bffffffffffffffffffffff166bffffffffffffffffffffff168152602001600982016000 90
54906101000a900460ff161515151581526020016009820160 0d9054906101000a900460ff161
5151515815260200160 0a820154815260200160 0b820154815250509050600080612 9ec8361
01000151613e42565b92509250506000612a108860008151811061 2a0357fe5b60200260200
10151613e42565b5050905060006002600 8b8152602001908152602001600020600901600c
9054906101000a900460ff16905060005b8951811015612d1357600080600061 2a698d858151
8110612a5c57fe5b602002602001015161 3e42565b92509250925085 67ffffffffffffff168367ffffffff
ffffff1614612adb576040517f08c379a0000000000000000000000000000000000000000000
0000000000081526004018080602001828103825260238152602001806 15a2360239139604
0019150506040518091039 0fd5b8415612b0957612b02816bffffffffffffffffffffffff168c61363c90919
063ffffffff16565b9a50612c9c565b8861010001518d8581518110612b1b57fe5b6020026020010
1511415612bde57612b8f612b80886bffffffffffffffffffffffff16612b728b6bffffffffffffffffffffffff168d610120
01516bffffffffffffffffffffff166136c490919063ffffffff16565b61374a90919063ffffffff16565b8d61363c
90919063ffffffff16565b9b50612bd7612bc88a61012001516bffffffffffffffffffffff16836bffffffffffffffffffffff
ff1661391b90919063ffffffff16565b8c61363c90919063ffffffff16565b9a50612c9b565b612c09896
1010001518e8681518110612bf357fe5b602002602001015161424890919063ffffffff16565b156

12c7657612c6f612c60886bfffffffffffffffffffffff16612c528b6bfffffffffffffffffffffff16856bfffffffffffffffffffff1
66136c490919063ffffffff16565b61374a90919063ffffffff16565b8d61363c90919063ffffffff16565b9
b50612c9a565b612c97816bfffffffffffffffffffffff168c61363c90919063ffffffff16565b9a505b5b5b856
7fffffffffffffff168e7f3f2c83616e48a2f8b3c9cc26499e2e5c61643714dac49027d82cf6732f96a91
a848460405180836bfffffffffffffffffffffff168152602001826bfffffffffffffffffffffff16815260200192505050
60405180910390a3505050808060010191505061 2a3f565b50612d208a898985614466565b
505050505050925092905056 5b816002600082815260200190815260200160002060020154 4
210612d9c576040517f08c379a0000000000000000000000000000000000000000000000000000000
0000000081526004018080602001828103825260328152602001806158636032913960400019
150506040518091039 0fd5b6000612da733611978565b90506000805b8451811015612f27576
000612df0868381518110612dc757fe5b602002602001015160016000 8a81526020019081526
020016000206 1455b90919063ffffffff16565b90508015612f19576000806000612e1989868151
8110612e0c57fe5b602002602001015161 3e42565b9250925092508667ffffffffffffff168367ffffffff
ffffff1614612e8b576040517f08c379a0000000000000000000000000000000000000000000000000
0000000000081526004018080602001828103825260238152602001806157f760239139604
0019150506040518091039 0fd5b612eac816bfffffffffffffffffffffff168761363c90919063ffffffff16565
b95508667ffffffffffffff168a7f7edae327fe79804b2f38bd490d874a0d188d3ae8911b303205abe4
7400086ac4848460405180836bfffffffffffffffffffffff168152602001826bfffffffffffffffffffffff1681526020
019250505060405180910390a35050505b5080806001019150506 12dad565b50612f8933826
002600089815260200190815260200160002060010160009054906101000a900473ffffffffffffffff
ffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff166146069092919063ffffffff16565b5050505050565b
60008060009054906101000a900473ffffffffffffffffffffffffffffffffffffffff16905090565b85600060026000
838152602001908152602001600020600301549050600081141581 5612fe65750804210155
b801561300a57506000801b60026000848152602001908152602001600020600801541 45b61
305f576040517f08c379a0000000000000000000000000000000000000000000000000000000000000
0081526004018080602001828103825260288152602001806 15a4660289139604001915050 6
040518091039 0fd5b506002600088815260200190815260200160002060090160 0d90549061
01000a900460ff166130d9576040517f08c379a0000000000000000000000000000000000000
0000000000000000000081526004018080602001828103825260288152602001806 1579e602
891396040019150506040518091039 0fd5b600186511480156130eb575060018551145b6131
40576040517f08c379a0000000000000000000000000000000000000000000000000000000000000
0815260040180806020018281038252602781526020018061595560279139604001915050 60
405180910390fd5b600061314b33611978565b90506131a861318382896000815181106131 6
157fe5b602002602001015189600081518110613173 7fe5b602002602001015161 38da565b6
00260008b815260200190815260200160002060070154614248909190 63ffffffff16565b6131fd
576040517f08c379a0000000000000000000000000000000000000000000000000000000000000 8
152600401808060200182810382526031815260200180615 7c66031913960400191505060 40
5180910390fd5b61320c8888888888888336146a8565b506132168861 1a0d565b50505050505
0505050565b60085481565b600866002600082815260200190815260200160002060030154

42106132985760405 17f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526022815260200180615b1760229139604001915050604051809103 90fd5b6132a78888888888833614 6a8565b91505096955050505050505 0565b6103e881565b60036020528060005260406000206000915054906101000a900473ffffffffffffffffffffffffffffffffffffffff1681565b60095481565b60008760026000828152602001908152602001600020600301544210613364576040517f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526022815260200180615b1760229139604001915050604051809103 90fd5b6133738989898989896146a8565b9150509796505050505050565b6133896141c7565b73ffffffffffffffffffffffffffffffffffffffff166000805490610100a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16146134495760405 17f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526020815260200180615 81a6026913960400191505060405180910390fd5b8073ffffffffffffffffffffffffffffffffffffffff166000805490610100a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff167f8be0079c531659141344cd1fd0a4f28419497f9722a3daafe3b4186f6b6457e060405160405180910390a3806000806101000a81548173ffffffffffffffffffffffffffffffffffffffff021916908373ffffffffffffffffffffffffffffffffffffffff1602179055505056 5b600460205280600052604060002060009150905080546001816001600116156101000203166002900480156136345780601f1061360957610100808354040283529160200191613634565b820191906000526020600020905b815481529060010190602001808311613617578290036 01f168201915b505050505081565b60008082840190508381101561 36ba576040517f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526 01b8152602001807f536166654d6174683a206164646974696f6e206f766572666c6f770000000000815250602001915050604051809103 90fd5b809150509291505056 5b600808314156136d75760009050613744565b60008284029050828482816136e857fe5b041461373f576040517f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526021815260200180615 9dc602191396040019150506040518091039 0fd5b809150505b929150505 65b600061378c838360405180604001604052806 01a81526020017f536166654d6174683a206469766973696f6e206279207a65726f0000000000008152506 14cc0565b905092915050565b61384f846323b872dd60e01b858585604051602401808473ffffffffffffffffffffffffffffffffffffffff168152602001837 3ffffffffffffffffffffffffffffffffffffffff168152602001828152602001935050505060405160208183030381529060405 2907bffffffffffffffffffffffffffffffffffffffffffffffffffffffff19166020820180517bffffffffffffffffffffffffffffffffffffffffffffffffffffffff838183161783525050505 0614d86565b50505050565b7fffffffffffffffffffffffffffffffff000000000000000000000000160001b8160000160006001600 01b8152602001908152602001600020819055506001600 01b8160010160007fffffffffffffffffffffffffffffffffffffffffffffffffffffffff00000000

0000000000000000160001b81526020019081526020016000208190555050565b6000816bffffff
ffffffffffffffffff166060846bffffffffffffffffffffffffff16901b60c08667ffffffffffffff16901b010160001b90509392
505050565b600061395d838360405180604001604052806 01e81526020017f536166654d617
4683a207375627472616374696f6e206f766572666c6f770000815250614e75565b905092915
050565b60006801000000000000000082106139c8576040517f08c379a00000000000000000
0000000000000000000000000000000000000000815260040180806020018281038252602 68
15260200180615 9fd60269139604001915050604051809103 90fd5b819050919050565b60008
073ffffffffffffffffffffffffffffffffffffff168273ffffffffffffffffffffffffffffffffffffff161415613a74576040517f08c379a
0000000000000000000000000000000000000000000000000000000000000000000000815260040180806020
018281038252601a8152602001807f43616e6e6f7420696e73657274207a65726f2061646472
65737300000000000081525060200191505060405180910390fd5b7fffffffffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffff67ffffffffffffff168367ffffffffffffff161415613b1e576040517f08c379a00000000000000
0000000000000000000000000000000000000000000000000815260040180806020018281038252602 60
188152602001807f43616e6e6f7420696e73657274206d61782075696e743634000000000000
0000815250602001915050604051809103 90fd5b60008460010160008473ffffffffffffffffffffffffffffffffffffff
ffffff1673ffffffffffffffffffffffffffffffffffff168152602001908152602001600020600090549061010 00a9
00467ffffffffffffff1667ffffffffffffff16141580613c065750600073ffffffffffffffffffffffffffffffffffff16846000
016000600186016 7ffffffffffffff1667ffffffffffffff168152602001908152602001600020600090549 0
6101000a900473ffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffff1614155b15613c145760
009050613cef565b818460000160006001860167ffffffffffffff1667ffffffffffffff1681526020019081
526020016000206000610 1000a81548173ffffffffffffffffffffffffffffffffffff021916908373ffffffffffffffffffff
ffffffffffffff160217905550600183018460010160008473ffffffffffffffffffffffffffffffffffff1673ffffffffffffffff
ffffffffffffffffffff168152602001908152602001600020600061010 00a81548167ffffffffffffff021916
908367ffffffffffffff160217905550600190505b9392505050565b60008083600101600 8473ffffff
ffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffff168152602001908152602001600020600090 5
4906101000a900467ffffffffffffff1667ffffffffffffff161415905092915050565b6000613d6f8383613
cf6565b613de1576040517f08c379a0000000000000000000000000000000000000000000000000
0000000000815260040180806020018281038252601b8152602001807f4d75737420686176
652041646472657373207 46f206765742049440000000008152506020019150506040518 09
10390fd5b60018360010160008473ffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffff16815
260200190815260200160002060009054906101000a900467ffffffffffffff160390509291505056
5b600806000 60c08460001c901c925060608460001c901c91508360001c905091939092505
65b60007fffffffffffffffffffffffffffffffffffffff0000000000000000000000160001b821415613ee6576040
517f08c379a0000000000000000000000000000000000000000000000000000000008152600
401808060200182810382526022815260200180615 8ba602291396040019150506040518091
0390fd5b60008360000160008481526020019081526020016000205490506000801b8114156
13f5c576040517f08c379a0000000000000000000000000000000000000000000000000000000
0008152600401808060200182810 38252602a8152602001806 1597c602a9139604001915050
60405180910390fd5b8091505092915050565b60006c0100000000000000000000082106

13fcd576040517f08c379a0000000000000000000000000000000000000000000000000000000000000815260040180806020018281038252602681526020018061577860269139604001915050604051809103 90fd5b81905091905056 5b60006140226103e86140146002600089815260200190815260200160020600a0154856bffffffffffffffffffffffff166136c490919063ffffffff16565b61374a90919063ffffffff16565b905060026000868152602001908152602001600020600901600c9054906101000a900460ff161561407f5761407a856140728385 6bffffffffffffffffffffffff1661363c90919063ffffffff16565b60008661446 5565b6141c0565b6000806140a160026000898152602001908152602001600020600901600c9054906101000a900460ff161561407f5761407a856140728385 6bffffffffffffffffffffffff1661391b90919063ffffffff16565b905060006141126141038 76bffffffffffffffffffffffff166140f585896136c490919063ffffffff16565b61374a90919063ffffffff16565b8361363c90919063ffffffff16565b9050600061415 7856bffffffffffffffffffffffff16614149866bffffffffffffffffffffffff168c6136c490919063ffffffff16565b61374a90919063ffffffff16565b90506141658a83838b614466565b6141ba8a61419c896bffffffffffffffffffffffff1661418e8d8b6136c490919063ffffffff16565b61374a90919063ffffffff16565b6000600a60009054906101000a900467ffffffffffffffff16614466565b50505050505b5050505050565b60003390509056 5b6000600160001b8214156141e6576000905061420556 5b6000801b8360000160008481526020019081526020016000205414159050 5b92915050565b6000806142188484614555b565b9050801561423e576000801b8460010160008581526020019081526020016000208190555505b809150509291505056 5b60008060008061425786613e42565b9250925092506000806000614266b88613e42565b925092509250614 2a0846bffffffffffffffffffffffff16836bffffffffffffffffffffffff166136c490919063ffffffff16565b6142cf826bffffffffffffffffffffffff16876bffffffffffffffffffffffff166136c490919063ffffffff16565b10156142e4576001965050505050505614460565b614313846bffffffffffffffffffffffff16836bffffffffffffffffffffffff166136c490919063ffffffff16565b614342826bffffffffffffffffffffffff16876bffffffffffffffffffffffff166136c490919063ffffffff16565b1115614357576000965050505050505614460565b816bffffffffffffffffffffffff16856bffffffffffffffffffffffff16101561438a576001965050505050505614460565b816bffffffffffffffffffffffff16856bffffffffffffffffffffffff1611156143bd576000965050505050505614460565b8267ffffffffffffffff168667ffffffffffffffff16141561442a576040517f08c379a0000000000000000000000000000000000000000000000000000000000000815260040180806020018281038252602d81526020018 0615a6e602d91396040019150506040518091039 0fd5b8267ffffffffffffffff168667ffffffffffffffff16101561445576001965050505050505614460565b600096505050505050505b92915050565b600061447c826005614f3590919063ffffffff16565b905060008411156144e9576144e88185600260008981526020019081526020016000206000016000905490 6101000a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff166146069092919063ffffffff16565b5b60008311156145545761455381846002600089815260200190815260200160002060010160009054906101000a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff166146069092919063ffffffff16565b5b5050505050565b60006145678383614 1cf565b6145745760009050614600565b60008360010160008481526020019081526020016000205490506000846000016000858152602001908152602001600020549050808560000160008481526020019081526020016000208190555081856001016000838152602001908152602001600020819055506000801b8560000160008681526020019081526020016000208190555060019250505 05b92915050565b6146a38363a9059cbb6

0e01b8484604051602401808373ffffffffffffffffffffffffffffffffffffff16815260200182815260200192505
0506040516020818303038152906040529 07bffffffffffffffffffffffffffffffffffffffffffffffffffffff191660208201
80517bffffffffffffffffffffffffffffffffffffffffffffffffffff838183161783525050505 0614d86565b505050565b6
00080600360008a81526020019081526020016000206000905490 6101000a900473ffffffffffffffff
ffffffffffffffffffff169050600073ffffffffffffffffffffffffffffffffffffff168173ffffffffffffffffffffffffffffffffffffff16146148a
d576319a05a7e60e01b7bffffffffffffffffffffffffffffffffffffffffffffffffffffffff19168173ffffffffffffffffffffffffffffffffffffff1
66319a05a7e858c89896040518563ffffffff1660e01b8152600401808573ffffffffffffffffffffffffffffffffffffff
f16815260200184815260200180602001828103825284848281815260200192508082843760
0081840152601f19601f820116905080830192505050955050505050506020604051808303 81
86803b1580156147df57600080fd5b505afa1580156147f3573d6000803e3d6000fd5b5050505
06040513d602081101561480957600080fd5b810190808051906020019092919050505 07bfffff
ffffffffffffffffffffffffffffffffffffffffffff1916146148ac576040517f08c379a000000000000000000000
0000000000000000000000000000000000000815260040180806020018281038252601f8152602
001807f75736572206e6f7420616c6c6f77656420746f20706c616365206f7264657200815250
602001915050604051809103 90fd5b5b506000806148d0600260008c8152602001908152602
00160002060040154613e42565b925092505060005b89518110156149c8576149238982815 1
81106148f057fe5b6020026020010 1516bffffffffffffffffffffff16836bffffffffffffffffffffff166136c4909190
63ffffffff16565b614965846bffffffffffffffffffffff168c84815181106149 4157fe5b6020026020010151
6bffffffffffffffffffffff166136c490919063ffffffff16565b106149bb576040517f08c379a00000000000
0000000000000000000000000000000000000000000000000000815260040180806020018281038252
6029815260200180615a9b6029913960400191505060405180910390fd5b80806001019150
506148d8565b505050600061 49d683611978565b9150600060026000 8b81526020019081526
020016000206005015490 5060005b8951811015614c4f5760008a8281518110614a0d57fe5b6
020026020010 1516bffffffffffffffffffffff1611614a79576040517f08c379a000000000000000000
0000000000000000000000000000000000000000000000000000815260040180806020018281038252602581526
020018061589560259139604001915050604051809103 90fd5b81898281518110614a8657fe
5b6020026020010 1516bffffffffffffffffffffff1611614b0f576040517f08c379a000000000000000000
0000000000000000000000000000000000000000000000000000815260040180806020018281038252600f81
526020018 07f6f7264657220746f6f20736d616c6c0000000000000000000000000000000008
1525060200191505060405180910390fd5b614b79614b43858c8481518110614b2257fe5b602
00260200101518c8581518110614b3657fe5b6020026020010 1516138da565b898381518110
614b4f57fe5b6020026020010 1516001600 08f81526020019081526020016000206150089092
919063ffffffff16565b15614c4257614bb2898281518110614b8d57fe5b6020026020010 1516bffff
ffffffffffffffff168461363c90919063ffffffff16565b92508367ffffffffffffffff168b7f9304f2fc7ed6d42c04
00e678dbc7283e1e9054889c3afea5f965adff66ef9eac8c8481518110614bed57fe5b60200260
200101518c8581518110614c0157fe5b6020026020010 1516040518 0836bffffffffffffffffffff1681
526020018 26bffffffffffffffffffffff168152602001925050506040518 0910390a35b8080600101915
0506149f6565b50614cb333308460026000 8f81526020019081526020016000206001016000 9
054906101000a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16613794909392

919063ffffffff16565b50509796505050505050565b6000808311829061 4d6c576040517f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382528381815181526020019150805190602001908083836000 5b83811015614d3157808201518184015260208101905061 4d16565b505050509050908101906 01f168015614d5e57808203805160018360200361 01000a031916815260200191505b509250505060405180910390fd5b506000838581614d7857fe5b0490508091505093925050565b6060614de8826040518060400160405280602081526020017f5361666545524332303a206c6f772d6c6576656c2063616c6c206661696c65648152508573ffffffffffffffffffffffffffffffffffffffff166152b69092919063ffffffff16565b905060008151111 5614e705780806020019051602081101 5614e0957600080fd5b810190808051906020019092919050505061 4e6f576040517f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526024815260200180615aed602a91396040019150506040518091039 0fd5b5b505050565b600083831115829061 4f22576040517f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382528381815181526020019150805190602001908083836000 5b83811015614ee75780820151818401526020810190506 14ecc565b5050505090509081019060 1f168015614f14578082038051600183602003610 1000a031916815260200191505b509250505060405180910390fd5b506000838503905080915050939250505056 5b6000614f4183836152ce565b614fb3576040517f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526 01b8152602001807f4d757374206861766520494420746f206765742041646472657373000000000008152506020019150506040518091039 0fd5b826000160006001840167ffffffffffffffff1667ffffffffffffffff16815260200190815260200160002060009054906101000a900473ffffffffffffffffffffffffffffffffffffffff169050592915050565b600806150148461 3e42565b9250505060006bffffffffffffffffffffffff16816bffffffffffffffffffffffff1614156150ab576040517f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526 01f8152602001807f496e73657274696e67207a65726f206973206e6f742073757070f72746564008152506020019150506040518091039 0fd5b600160001b84141580156150e257507fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff0000000000000000000000016000 1b8414155b615154576040517f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526 01e8152602001807f496e73657274696e6720656c656d656e74206973206e6f742076616c69640000815250602001915050604051809103 90fd5b61515e85856141cf565b1561516d576000915050615 2af565b600160001b8314158015615198575060008 01b8560010160008581526020019081526020016000205414 5b156151a757600091505061 52af565b6151ba848461424890919063ffffffff16565b6151c8576000915050615 2af565b5b6000801b8560000160008581526020019081526020016000205414156152065784600101600084815260200190815260200160002054925061 51c9565b6000808490505b8091508660000160008281526020019081526020016000205490506152361424890919063ffffffff16565b61520d57858760000160008481526020019081526020016000208190555085876001016000838152602001908152602001600020819055508187600101600088815260200190815260200160002081905550808760000160008881526020019081526020016000208

1905550600193505050505b9392505050565b60606152c5848460008561535456 5b905093925050505 65b60008073ffffffffffffffffffffffffffffffffffffffff16836000016000600185016 7ffffffffffffff1667ffffffffffffff1681526020019081526020016000206000905490610 1000a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16141590509291505056 5b606082471015615 3af576040517f08c379a0000000000000000000000000000000000000000000000000000000081526004018060200182810382526026815260200180615 8dc60269139604001915050604051809103 90fd5b6153b8856154fd565b61542a576040517f08c379a0000000000000000000000000000000000000000000000000000000081526004018080602001828103825260 1d8152602001807f416464726573733a2063616c6c20746f206e6f6e2d636f6e74726163740000008152506020019150506040518091039 0fd5b600060608673ffffffffffffffffffffffffffffffffffffffff16858760405180828051906020019080838 35b6020831061547a57805182526020820191506020810190506020830392506154575 65b60018360200361010 00a038019825116818451168082178552505050505050905001915050600060405180830381858 75af1925050503d8060008114615 4dc57604051915060 1f19603f3d011682016040523d82523d6000602084013e6154e1565b606091505b5091509150 6154f1828286615510565b92505050949350505050565b600080823b905060008111915050919050565b606083156155205782905061 55d5565b60008351115615533578251808460200 1f5b816040517f08c379a0000000000000000000000000000000000000000000000000000000081526004018080602001828103825283818151815260200191508051906020019080838360005b838110156155 9a57808201518184015260208101905061557f565b505050509050908101906 01f1680156155c75780820380516001836020036101000a031916815260200191505b5092505050604051809103 90fd5b9392505050565b828054600181600116156101000203166002900490600052602060002090601f016020900481019282601f1061561d57805160ff1916838001178555 61564b565b828001600101855582156 1564b579182015b828111156 1564a57825182559160200191906001019061562f565b5b509050615658919061575a565b5090565b5080546001816001161561010002031660029004600082558060 1f106156825750615 6a1565b601f016020900490600052602060002090810190615 6a0919061575a565b5b50565b604051806101c0016040528060007 3ffffffffffffffffffffffffffffffffffffffff16815260200160007 3ffffffffffffffffffffffffffffffffffffffff16815260200160008152602001600081526020016000815260200160008019168152602001600080191681526020016000815260200160008019168152602001600080191681526020016000 6bffffffffffffffffffffffff16815260200160001515815260200160001515815260200160008152602001600081525090565b5b8082111561577357600081600090555060010161575b565b509056fe5361666543617374 3a2076616c756520646f65736e27742066697420696e203936206269747 36e6f7420616c6c6f77656420746f20736574746c65206175637469 6f6e2061746f6d963616c6c7970726563616c63756c61746553656c6c416d6f756e7453756d20697320616c726561647920746f6f20616476616e6365644f6e6c792074686520757365722063616e2063616e6e3656c20686973206f72646572734f776e61626c653a206e6577206f776e65722069732074686 5207a65726f206164726573737 46f6b656e732063616e6e6f742062652061756374696f6e656420666f7220667265656e6f206c6f6e67657220696e206f7264657220706c6163656d656e7420616e642063616e63656c6174696f6e2070686173655f6d696e427579416d6f756e7473206d7

5737420626520677265617465722074686 16e2030547279696e6720746f20676574206e6578
74206f66206c61737420656c656d656e7441646472657373 3a20696e7375666669636965 6e7
42062616c616e636520666f722063616c6c466565206973206e6f7420616c6c6f77656420746f
20626520736574206869676865722074686 16e20312e352561756374696f20656e6420646
17465206d7573742062652069 6e207468652066757475726 54f6e6c79206f6e65206f726465
72206361 6e20626520706c616365642061746f6d6963616c6c79547279696e6720746f2067657
4206e657874206f66206e6f6e2d6578697374656e7420656c656d656e746d696e696d756d426
96464696e67416d6f756e745065724f72646572206973206e6f7420616c6c6f77656420746f20
6265207a65726f536166654d6174683a206d756c7469706c69636174696f6e206f766572666c
6f7753616665436173743a2076616c756520646f65736e27742066697420696e203634206269
74736f6e6c7920616c6c6f77656420746f20636c61696d20666f722073616d652075736572417
56374696f6e206e6f7420696e20736f6c7574696f6e207375626d697373696f6e207068617365
7573657220697 3206e6f7420616c6c6f77656420746f20706c6163652073616d65206f726465
722074776963656c696d6974207072696365206e6f74206265747465722074686 16e206d696d
696d616c206f6666657274696d6520706572696f647320617265206e6f7420636f6e666967757
2656420636f72726563746c795361666545524332303a204552433230206f7065726174696f6
e20646964206e6f74207375636365646e6f206c6f6e67657220696e206f7264657220706c61
63656d656e74207068617365a2646970667358221220 9d3f3585da1ffe9756c8a2e5f28991662
e7f098068faf72dd2ec014b6bd536c964736f6c634300060c0033", "deployedBytecode":
"0x6080604052348015610010576000 80fd5b506004361061015857600035 60e01c80637882d
eaf116100c3578063d73792a91161007c578063d73792a914610ccf578063e4a59ef414610ced
578063e86dea4a14610d45578063ec20d0bb14610d63578063f2fde38b14610fe0578063f59c2
f0614611024576101585 65b80637882deaf146106425780637ed18b701461071f5780638da5c
b5b146107e157806391cfc1d414610815578063a7e7664414610a54578063d225269c14610a
7257610158565b80633e12905f116101155780633e12905f1461046157806340b20b09146104
9957806355fc62d2146104e75780635cefb291146105c257806363c699a4146105ea57806371
5018a6146106385761015856 5b80630a4cd6c91461015d57806315d37b4b146102f15780631
9a50f4914610333578063199d5cd1461035b5780632b956ff7146103bd5780632e9936111461
041f575b600080fd5b6102db6004803603610160811015610174576000 80fd5b810190808035
73ffffffffffffffffffffffffffffffffffffffff1690602001909291908035 73ffffffffffffffffffffffffffffffffffffffff16906020019
092919080359060200190929190803590602001909291908035 6bffffffffffffffffffffffff1690602001
909291908035 6bffffffffffffffffffffffff1690602001909291908035906020019092919080359060200
19092919080351515906020019092919080357 3ffffffffffffffffffffffffffffffffffffffff16906020019092919
080359060200190640100000000811115610255576000 80fd5b820183602082011115610267 5
7600080fd5b803590602001918460018302840111640100000000831117156102895760008 0f
d5b91908080601f016020809104026020016040519081016040528093929190818152602001
838380828437600081840152601f19601f8201169050808301925050505050505091929192 90
5050506110cb565b604051808281526020019150506040518091039 0f35b61031d600480360
360208110156103075760008 0fd5b810190808035906020019092919050505061178a565b60

405180828152602001915050604051809103 90f35b61033b6117e4565b604051808267ffffffffffff
ffff16815260200191505060405180910390f35b61039d60048036036020811015610157600
080fd5b81019080803573ffffffffffffffffffffffffffffffffffffffff16906020019092919050505061 17fe565b60
4051808267ffffffffffffffff16815260200191505060405180910390f35b6103ff60048036036020811
0156103d357600080fd5b81019080803573ffffffffffffffffffffffffffffffffffffffff16906020019092919050505
0611978565b604051808267ffffffffffffffff16815260200191505060405180910390f35b61044b60
0480360360208110156104357600080fd5b810190808035906020019092919050505061 1a0
d565b604051808281526020019150506040518091039 0f35b610497600480360360408110156
10477576000 80fd5b810190808035906020019092919080359060200190929190505050611f
b4565b005b6104e56004803603604081101561 04af57600080fd5b810190808035906020019 0
929190803573ffffffffffffffffffffffffffffffffffffffff16906020019092919050505061 22d6565b005b610513
60048036036020811015610 4fd57600080fd5b810190808035906020019092919050505061 24
34565b604051808f73ffffffffffffffffffffffffffffffffffffffff1681526020018e73ffffffffffffffffffffffffffffffffffffffff1681
526020018d81526020018c81526020018b81526020018a8152602001898152602001888152 6
020018781526020018 66bffffffffffffffffffffff16815260200185151581526020184151581526020
018381526020018281526020019e505050505050505050505050505060405180910390f35
b6105ca612512565b604051808267ffffffffffffffff16815260200191505060405180910390f35b610
62060048036036040811015610600 57600080fd5b810190808035906020019092919080359 0
602001909291905050506125 2c565b604051808215158152602001915050604051809103 90f
35b61064061255b565b005b6107026004803603604081101561065857600080fd5b81019080
803590602001909291908035906020019064010000000081111561067f57600080fd5b820183
602082011115610691576000 80fd5b803590602001918460208302840111640100000000831 1
17156106b357600080fd5b919080806020026020016040519081016040528093929190818 15
260200183836020028082843760008184015260 1f19601f820116905080830192505050505505
050919291929050505061 26e1565b604051808381526020018281526020019250505060405 1
80910390f35b6107df60048036036040811015610735 7600080fd5b810190808035906020 01
909291908035906020019064010000000081111561075c57600080fd5b820183602082011115
61076e57600080fd5b803590602001918460208302840111640100000000831 1171561079057
600080fd5b919080806020026020016040519081016040528093929190818152602001838 36
020028082843760008184015260 1f19601f820116905080830192505050505050919291929
0505050612d2d565b005b6107e9612f90565b604051808273ffffffffffffffffffffffffffffffffffffffff16815260
200191505060405180910390f35b610a52600480360360a081101561082b57600080fd5b8101
908080359060200190929190803590602001906401000000081111561085257600080fd5b82
018360208201111561086457600080fd5b803590602001918460208302840111640100000000
831117156108657600080fd5b919080806020026020016040519081016040528093929190818
181526020018383602002808284376000818401526 01f19601f820116905080830192505050505
050505091929192908035906020019064010000000081111561 08e657600080fd5b820183602
082011115610 8f857600080fd5b803590602001918460208302840111640100000000831117 15
61091a57600080fd5b919080806020026020016040519081016040528093929190818152602

00183836020028082843760008184015260 1f19601f8201169050808301925050505050509
19291929080359060200190640100000000811115610 97a57600080fd5b82018360208201111
561098c57600080fd5b80359060200191846020830284011164010000000831117156109ae5
7600080fd5b91908080602002602001604051908101604052809392919081815260200183
60200280828437600081840152601f19601f8201169050808301925050505050505091929192
90803590602001906401000000000811115610a0e57600080fd5b820183602082011115610a20
57600080fd5b80359060200191846001830284011164010000000083111715610a4257600080
fd5b909192939192939050505061 2fb9565b005b610a5c613221565b60405180828152602001
91505060405180910390f35b610caf600480360360a0811015610a8857600080fd5b81019080
80359060200190929190803590602001906401000000000811115610aaf57600080fd5b820183
602082011115610ac157600080fd5b80359060200191846020830284011164010000000083111
1715610ae357600080fd5b91908080602002602001604051908101604052809392919081815
26020018383602002808284376000818401526 01f19601f8201169050808301925050505050
05091929192908035906020019064010000000811115610b4357600080fd5b8201836020820
11115610b5557600080fd5b80359060200191846020830284011164010000000083111715610
b7757600080fd5b91908080602002602001604051908101604052809392919081815260200 1
838360200280828437600081840152601f19601f8201169050808301925050505050509192
91929080359060200190640100000000811115610bd757600080fd5b8201836020820111156 1
0be957600080fd5b80359060200191846020830284011164010000000083111715610c0b5760
0080fd5b91908080602002602001604051908101604052809392919081815260200183836 02
00280828437600081840152601f19601f8201169050808301925050505050509192919290 8
03590602001906401000000000811115610c6b57600080fd5b8201836020820111156 10c7d576
00080fd5b80359060200191846001830284011164010000000083111715610c9f57600080fd5b
909192939192939050505061 3227565b604051808267ffffffffffffffff168152602001915050604 05
180910390f35b610cd76132b4565b604051808281526020019150506040518091039 0f35b610
d19600480360360208110156 10d0357600080fd5b810190808035906020019092919050505 0
6132ba565b604051808273ffffffffffffffffffffffffffffffffffffffff168152602001915050604 05180910390f35
b610d4d6132ed565b604051808281526020019150506040518091039 0f35b610fc0600480360
360c0811015610d7957600080fd5b810190808035906020019092919080359060200190640 1
0000000811115610da057600080fd5b8201836020820 11115610db257600080fd5b80359060
200191846020830284011164010000000083111715610dd457600080fd5b9190808060200260
200160405190810160405280939291908181526020018383602002808284376000818401526
01f19601f8201169050808301925050505050509192919290803590602001906401000000 0
0811115610e3457600080fd5b8201836020820 11115610e4657600080fd5b803590602001918
460208302840111640100000000831117 15610e6857600080fd5b91908080602002602001604
051908101604052809392919081815260200183836020028082843760008184015260 1f1960
1f8201169050808301925050505050509192919290803590602001906401000000000811115
610ec857600080fd5b8201836020820 11115610eda57600080fd5b803590602001918460208 3
028401116401000000083111715610efc57600080fd5b91908080602002602001604051908 1

01604052809392919081815260200183836020028082843760008184015260 1f19601f820116
90508083019250505050505050919291929080359060200190640100000000811115610f5c57
600080fd5b8201836020820111156 10f6e57600080fd5b80359060200191846001830284011 16
401000000008311171561 0f9057600080fd5b909192939192939080357 3ffffffffffffffffffffffffffffffffffffff
fff16906020019092919050505061 32f3565b60405180826 7ffffffffffffffff16815260200191505060
405180910390f35b6110226004803603602081101561 0ff657600080fd5b81019080803573fffffff
ffffffffffffffffffffffffffffffff16906020019092919050505061 3381565b005b6110506004803603602 08
1101561103a57600080fd5b810190808035906020019092919050505061358c565b60405180
80602001828103825283818151815260200191508051906020019080838360005b838110156
110905780820151818401526020810190506110755 65b505050509050908101906 01f168015
6110bd57808203805160018360200361010 00a031916815260200191505b509250505060405
180910390f35b60006111443330611 11d6103e861110f6110f26009546103e861363c90919063f
fffffff16565b8e6bffffffffffffffffffffffff166136c490919063fffffff16565b61374a90919063fffffff16565b8
f73ffffffffffffffffffffffffffffffffffffff166137949093929190 63fffffff16565b6000886bffffffffffffffffffffff16116
111c8576040517f08c379a0000000000000000000000000000000000000000000000000000000
0008152600401808060200182810382526 01a8152602001807f63616e6e6f742061756374696
f6e207a65726f20746f6b656e73000000000000815250602001915050604051809103 90fd5b60
0876bffffffffffffffffffffff161161122f576040517f08c379a0000000000000000000000000000000
0000000000000000000000008152600401808060200182810382526023815260200180615 8
406023913960400191505060405180910390fd5b60008611611288576040517f08c379a00000
0000000000000000000000000000000000000000000000008152600401808060200182 8
10382526036815260200180615 9a660369139604001915050604051809103 90fd5b888a1115
6112e1576040517f08c379a0000000000000000000000000000000000000000000000000000000
0000081526004018080602001828103825260298152602001806 15ac4602991396040019150
5060405180910390fd5b428911611339576040517f08c379a0000000000000000000000000000
0000000000000000000000008152600401808060200182810382526026815260200180 18
061592f602691396040019150506040518091039 0fd5b61134f60016008546 1363c90919063fff
fffff16565b60088190555061137260016000600854815260200190815260200160002061385 5
565b600061137d33611978565b905060405180610 1c001604052808e73ffffffffffffffffffffffffffffffff
fff1681526020018d73ffffffffffffffffffffffffffffffffffffffff1681526020018c81526020018b8152602001611 3
3da838b8d6138da565b8152602001888152602001600081526020016001600160001b8152602001
6000801b81526020016000 6bffffffffffffffffffffffff1681526020016000151581526020018615158 15
260200160095481526020018781525060026000600854815260200190815260200160002060
0082015181600001600006101000a81548173ffffffffffffffffffffffffffffffffffffffff02191690837 3ffffffffffffffff
ffffffffffffffffff160217905550602082015181600101600061010 00a81548173ffffffffffffffffffffffffffffffff
fffffff021916908373ffffffffffffffffffffffffffffffffffffffff160217905550604082015181600201556060820 1
518160030155608082015181600401556 0a08201518160050155 60c08201518160060155 60e
08201518160070155610100820151816008015561012082015181600901600061010 00a8154
816bffffffffffffffffffffff02191690836bffffffffffffffffffffff160217905550610140820151816009016 00c

6101000a81548160ff021916908315150217905550610160820151816009016006d6101000a81
548160ff021916908315150217905550610180820151816006a01556101a082015181600b0155
905050836003600060085481526020019081526020016000206000610100 0a81548173ffffffffff
ffffffffffffffffffffffffff021916908373ffffffffffffffffffffffffffffffffffffff160217905550826004600060085481
526020019081526020016000209080519060200190611 62b9291906155dc565b508b73ffffffffff
ffffffffffffffffffffffffff168d73ffffffffffffffffffffffffffffffffffffff166008547f728d0fed13687f1840de94f5ae640
eae49b43eda26b7ffe97e55e79a0de6e40c8e8e868f8f8f8f8e8e604051808a815260200189815
26020018867ffffffffffffff168152602001876bffffffffffffffffffffff168152602001866bffffffffffffffffffffff16
815260200185815260200184815260200183 73ffffffffffffffffffffffffffffffffffffff168152602001806020
018281038252838181518152602001915080519060200190808383600 05b838110156117335
780820151818401526020810190506117185 65b505050509050908101906 01f168015611760
578082038051600183602003610100 0a031916815260200191505b509a50505050505050505
050506040518091039 0a46008549150509b9a505050505050505050505050565b600042600260
008481526020019081526020016000206003015410156117b357600090506117df565b6117dc
426002600085815260200190815260200160002060030154613 91b90919063ffffffff16565b905
05b919050565b60076000905490610100 0a900467ffffffffffffffff1681565b600061183c611837600
160076000905490610100 0a900467ffffffffffffffff1667ffffffffffffffff1661363c90919063ffffffff16565b6
13965565b600760006101000a81548167ffffffffffffffff021916908367ffffffffffffffff160217905550611
890600760009054906101000a900467ffffffffffffffff168360056139d09092919063ffffffff16565b611
902576040517f08c379a0000000000000000000000000000000000000000000000000000000000
008152600401808060200182810382526017815260200180 7f5573657220616c72656164792
072656769737465726564000000000000000008152506020019150506040518091039 0fd5b
60076000905490610100 0a900467ffffffffffffffff1690508173ffffffffffffffffffffffffffffffffffffff167f6838f67c
b358c332087b73dbe769a6c869a5f87225236c16dda5d0013a77074c82604051808267ffffffffffff
ffff16815260200191505060405180910390a2919050565b600061198e826005613cf69091906
3ffffffff16565b156119ae576119a7826005613d6390919063ffffffff16565b9050611a08565b6119b
7826117fe565b90508173ffffffffffffffffffffffffffffffffffffff168167ffffffffffffff167f969d438b19b6b5fb3c0
d3c6f16867e519fcec4233799d6be29d1c00f2045ba266040516040518091039 0a35b9190505
65b600081600060026000838152602001908152602001600020600301549050600081141580
15611a3c5750804210155b8015611a6057506000801b60026000848152602001908152602001
6000206008015414 5b611ab5576040517f08c379a0000000000000000000000000000000000
000000000000000000000815260040180806020018281038252602881526020018061 5a4
660289139604001915050604051809103 90fd5b506000806000611ada600260008881526020
019081526020016000206004015461 3e42565b92509250925060006002600088815260200190
815260200160002060060154905060006002600089815260200190815260200160002060070
154905060008060008590505b6000611b4685600160008f8152602001908152602001600020
0613e6890919063ffffffff16565b90507ffffffffffffffffffffffffffffffffffffffff0000000000000000000000160
001b811415611b795750611c02565b809450611b8585613e42565b909150816bffffffffffffffffffffff
169150806bffffffffffffffffffffffff169050809450819550505061 1bc5838761363c90919063ffffffff1656

5b955050611be982876bfffffffffffffffffffffff166136c490919063fffffff16565b611bfc84876136c4909
19063fffffff16565b10611b1f575b60008511801561 1c445750611c2e82876bfffffffffffffffffffff1661
36c490919063fffffff16565b611c4184876136c490919063fffffff16565b10155b15611d4d576000
611c90611c8185611c73868b6bfffffffffffffffffffff166136c490919063fffffff16565b61374a9091906
3fffffff16565b8761391b90919063fffffff16565b9050808310611d1b576000611cae828561391b
90919063fffffff16565b9050611cb981613f66565b600260008f81526020019081526020016000
20600901600061010 00a8154816bfffffffffffffffffffffff02191690836bfffffffffffffffffffff160217905550
611d10828861391b90919063fffffff16565b9650859b5050611d47565b611d2e838761391b909
19063fffffff16565b9550611d44600088611d3f89613f66565b6138da565b9a505b50611dd9565
b866bfffffffffffffffffffff16851115611d7e57611d7600087611d7288613f66565b6138da565b9950
611dd8565b611d8a600087896138da565b9950611dd5611dd0886bfffffffffffffffffffff16611dc289
6bfffffffffffffffffffff16896136c490919063fffffff16565b61374a90919063fffffff16565b613f66565b9
0505b5b89600260008d81526020019081526020016000206008018190555084600260008d81
52602001908152602001600020600b01541115611e42576001600260008d815260200190815
26020016000206009016 00c6101000a81548160ff0219169083151502179055505b611e5c8b8
26bfffffffffffffffffffff168a89613fd5565b8a7f4d160a2a345f2faeb9ac2e65272820b8ca5473b80aa
bef416bdf7e07ee7f591082878d60405180846bfffffffffffffffffffff168152602001836bfffffffffffffffffff
ff16815260200182815260200193505050506040518091039 0a26000600360008d8152602001
908152602001600020600061010 00a81548173fffffffffffffffffffffffffffffffffffff02191690837 3fffffffffffff
fffffffffffffffffffff160217905550600460008c81526020019081526020016000206000611f32919
061565c565b6000801b600260008d81526020019081526020016000206004018190555060000
801b600260008d81526020019081526020016000206007018190555060006002600 8d81526
020019081526020016000206006018190555060006002600 8d8152602001908152602001600
0020600501819055505050505050505050919050565b816000600260008381526020019081
526020016000206003015490506000811415801 5611fe15750804210155b80156120055750
6000801b600260008481526020019081526020016000206008015414 5b61205a576040517f0
8c379a0000000000000000000000000000000000000000000000000000000081526004018
0806020018281038252602881526020018061 5a46602891396040019150506040518091039 0f
d5b50600061207c600260008681526020019081526020016000206004015461 3e42565b9250
5050600060026000868152602001908152602001600020600601549050600060026000878 15
26020019081526020016000206007015490506000 5b858110156121285761 20e68260016000
8a81526020019081526020016000206 13e6890919063fffffff16565b915060006120f383613e4
2565b9250505061211881 6bfffffffffffffffffffffff168561363c90919063fffffff16565b9350508080600
1019150506120b9565b507fffffffffffffffffffffffffffffffffffffffff0000000000000000000000160001b8114
156121c2576040517f08c379a0000000000000000000000000000000000000000000000000000
0000081526004018080602001828103825260198152602001807f7265616368656420656e
64206f66206f72646572206c6973740000000000000815250602001915050604051809103 90
fd5b6000806121ce83613e42565b9250925050612028 16bfffffffffffffffffffff16866bfffffffffffffffffff
ff166136c490919063fffffff16565b612223836bfffffffffffffffffffff16866136c490919063fffffff16565

b10612296576040517f08c379a0000000000000000000000000000000000000000000000000
0000000815260040180806020018281038252601981526020018 07f746f6f206d616e79206f7
2646572732073756d6d656420757000000000000000815250602001915050604051809103 90
fd5b83600260008a8152602001908152602001600020600601819055508260026000 8a81526
0200190815260200160002060070181905550505050505050505050565b6122de6141c7565b73
ffffffffffffffffffffffffffffffffffffffff16600080549061010 00a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffff
ffffffffffffffffffffff161461239e576040517f08c379a0000000000000000000000000000000000000
0000000000000000000815260040180806020018281038252602081526020018 07f4f776e616
26c653a2063616c6c6572206973206e6f7420746865206f776e657281525060200191505060 4
05180910390fd5b600f8211156123f8576040517f08c379a0000000000000000000000000000000000000
0000000000000000000000000000815260040180806020018281038252602d8152602001806
15902602d91396040019150506040518091039 0fd5b6124018161197856 5b600a6000610100
0a81548167ffffffffffffffff021916908367ffffffffffffffff160217905550816009819055505050565b6002
602052806000526040600020600091509050806000016000905490610100 0a900473ffffffffffffffff
ffffffffffffffffffffff16908060010160009054906101000a900473ffffffffffffffffffffffffffffffffffffffff16908060
020154908060030154908060040154908060050154908060060154908060070154908060080
154908060090160009054906101000a90046bffffffffffffffffffffffff16908060090160 0c90549061010
00a900460ff16908060090160 0d9054906101000a900460ff169080600a01549080600b015490
508e565b600a60009054906101000a900467ffffffffffffffff1681565b600612553826001600 0868
152602001908152602001600020 6141cf90919063ffffffff16565b905092915050565b61256361
41c7565b73ffffffffffffffffffffffffffffffffffffffff16600080549061010 00a900473ffffffffffffffffffffffffffffffffffffffff16
73ffffffffffffffffffffffffffffffffffffffff161461262357604051 7f08c379a0000000000000000000000
0000000000000000000000000000000815260040180806020018281038252602081526020018 0
7f4f776e61626c653a2063616c6c6572206973206e6f7420746865206f776e657281525060200
191505060405180910390fd5b600073ffffffffffffffffffffffffffffffffffffffff16600080549061010 00a900473
ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff167f8be0079c531659141344cd1fd0a4f28
419497f9722a3daafe3b4186f6b6457e0604051604051809103 90a3600080600061010 00a815
48173ffffffffffffffffffffffffffffffffffffff021916908373ffffffffffffffffffffffffffffffffffffffff160217905550565b6000
80836000801b60026000838152602001908152602001600020600801541415612774576040 5
17f08c379a0000000000000000000000000000000000000000000000000000000081526004
0180806020018281038252601881526020018 07f41756374696f6e206e6f7420796574206669
6e697368656400000000000000008152506020019150506040518091039 0fd5b60005b84518
11015612837576127b885828151811061278f57fe5b602002602001015160016000898152602
0019081526020016000206142 0b90919063ffffffff16565b61282a576040517f08c379a0000000
0000000000000000000000000000000000000000000000000000000081526004018080602001828 10
38252601c8152602001807f6f72646572206973206e6f206c6f6e67657220636c61696d61626c
650000000081525060200191505060405180910390fd5b8080600101915050612777565b506
128406156a4565b60026000878152602001908152602001600020604051806101c001604052
90816000820160009054906101000a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff

fffff1673ffffffffffffffffffffffffffffffffff16815260200160018201600090549061010 00a900473fffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffff16815260200160028201548152602001600382015481526020016004820154815260200160058201548152602001600682015481526020016007820154815260200160088201548152602001600982016000905490610100 0a90046bfffffffffffffffffffffff166bfffffffffffffffffffff166bfffffffffffffffffffff16815260200160098201600c9054906101000a900460ff161515151581526020016009820160 0d9054906101000a900460ff161515151581526020016009820160 0d9054906101000a900460ff161515151581526020016009820160 0a8201548152602001600b8201548152505090506000806129ec836101000151613e42565b92509250506000612a108860008151811061 2a0357fe5b6020026020010151613e42565b505090506000600260008b8152602001908152602001600020600901600c9054906101000a900460ff16905060005b8951811015612d13576000806000612a698d8581518110612a5c57fe5b6020026020010151613e42565b92509250925085677fffffffffffff168367ffffffffffffff1614612adb576040517f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526023815260200180615a23602391396040019150506040518091039 0fd5b8415612b0957612b02816bfffffffffffffffffffff168c61363c90919063fffffff16565b9a50612c9c565b8861010001518d8581518110612b1b57fe5b6020026020010151141561 2bde57612b8f612b80886bfffffffffffffffffffff16612b728b6bfffffffffffffffffffff168d61012001516bfffffffffffffffffffff166136c490919063fffffff16565b61374a90919063fffffff16565b8d61363c90919063fffffff16565b9b50612bd7612bc88a61012001516bfffffffffffffffffffff16836bfffffffffffffffffff1661391b90919063fffffff16565b8c61363c90919063fffffff16565b9a50612c9b565b612c098961010001518e8681518110612bf357fe5b6020026020010151614248909190 63ffffffff16565b15612c7657612c6f612c60886bfffffffffffffffffffff16612c528b6bfffffffffffffffffffff16856bfffffffffffffffffff166136c490919063fffffff16565b61374a90919063fffffff16565b8d61363c90919063fffffff16565b9b50612c9a565b612c97816bfffffffffffffffffffff168c61363c90919063fffffff16565b9a505b5b5b8567fffffffffffff168e7f3f2c83616e48a2f8b3c9cc26499e2e5c61643714dac49027d82cf6732f96a91a848460405180836bfffffffffffffffffffff168152602001826bfffffffffffffffffffff168152602001925050506040518091039 0a3505050808060010191505061 2a3f565b50612d208a898985614466565b505050505050925092905056 5b8160026000828152602001908152602001600020600201544210612d9c576040517f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526032815260200180615863603291396040019150506040518091039 0fd5b6000612da73361197856 5b90506000805b8451811015612f27576000612df0868381518110612dc757fe5b6020026020010151600160008a81526020019081526020016000206145 5b90919063fffffff16565b90508015612f1957600080600612e19898681518110612e0c57fe5b6020026020010151613e42565b92509250925086677fffffffffffff168367ffffffffffffff1614612e8b576040517f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526023815260200180615 7f76023913960400191505060405180910390fd5b612eac816bfffffffffffffffffffff168761363c90919063fffffff16565b95508667ffffffffffffff168a7f7edae327fe79804b2f38bd490d874a0d188d3ae8911b303205abe47400086ac484846040518083 6bfffffffffffffffffffff168152602001826bfffffffffffffffffffff168152602001925050506040518091039 0a35050505b508080600101915050612dad565b5

0612f893382600260008981526020019081526020016000206001016000905490610100 0a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16614606909291906 3ffffffff16565b5050505050565b600080600090549061010000a900473ffffffffffffffffffffffffffffffffffffffff16905090565b8560006002600083815260200190815260200160002060030154905060008114158015612fe657508042101 55b801561300a57506000801b600260008481526020019081526020016000206080154145b61305f576040517f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526028815260200180615a466028913960400191505060405180910390fd5b506002600088815260200190815260200160002060090160 0d9054906101000a900460ff166130d9576040517f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526028815260200180615 79e602891396040019150506040518091039 0fd5b600186511480156130eb57506001855114 5b613140576040517f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526027815260200180615955602791396040019150506040518091039 0fd5b600061314b33611978565b90506131a8613183828960008151811061316157fe5b602002602001015189600081518110613161 7657fe5b602002602001015160138da565b600260008b81526020019081526020016000206007015461424890919063ffffffff16565b6131fd576040517f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526031815260200180615 7c66031913960400191505060405180910390fd5b61320c888888888888336146a8565b5061321688611a0d565b505050505050505050565b60085481565b600086600260008281526020019081526020016000206030154421061329857604051 7f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526022815260200180615b1760229139604001915050604051809103 90fd5b6132a788888888888336146a8565b9150509695505050505050565b6103e881565b600360205280600052604060002060009150549061010000a900473ffffffffffffffffffffffffffffffffffffffff1681565b60095481565b600087600260008281526020019081526020016000206030154421061336457604051 7f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526022815260200180615b176022913960400191505060405180910390fd5b6133738989898989896146a8565b9150509796505050505050565b6133896141c7565b73ffffffffffffffffffffffffffffffffffffffff16600080549061010000a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff161461344957604 0517f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526020815260200180 7f4f776e61626c653a2063616c6c6572206973206e6f7420746865206f776e657281525060200191505060405180910390fd5b600073ffffffffffffffffffffffffffffffffffffffff168173ffffffffffffffffffffffffffffffffffffffff1614156134cf576040517f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526026815260200180615 81a6026913960400191505060405180910390fd5b8073ffffffffffffffffffffffffffffffffffffffff16600080549061010000a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff167f8be0079c531659141344cd1fd0a4f28419497f9722a3daafe3b4186f6b6457e0604051605 04051809103 90a3806000806101000a81548173ffffffffffffffffffffffffffffffffffffffff021916908373ffffffffff

fffffffffffffffffffffffff160217905550505656b60046020528060005260406000206000915090508054600181600116156101000203166002900480601f01602080910402602001604051908101604052809291908181526020018280546001816001161561010002031660029004801561363457806001f1061360957610100808354040283529160200191613634565b820191906000526020600020905b81548152906001019060200180831161361757829003601f168201915b505050505081565b60008082840190508381101561368a576040517f08c379a000000000000000000000000000000000000000000000000000000000815260040180806020018281038252601b8152602001807f536166654d6174683a206164646974696f6e206f766572666c6f770000000000815250602001915050604051809103901fd5b8091505092915050565b60008083141561368d7576000905061374565b60008284029050828482816136e857fe5b041461373f576040517f08c379a000000000000000000000000000000000000000000000000000000000815260040180806020018281038252602181526020018061590dc602191396040019150506040518091039fd5b809150505b92915050565b600061378c838360405180604001604052806001a81526020017f536166654d6174683a206469766973696f6e206279207a65726f0000000000000815250614cc0565b905092915050565b61384f846323b872dd60e01b858585604051602401808473ffffffffffffffffffffffffffffffffffffffffffffffff1681526020018373ffffffffffffffffffffffffffffffffffffffff168152602001828152602001935050505050604051602081830303815290604052907bffffffffffffffffffffffffffffffffffffffffffffffffffffffff19166020820180517bffffffffffffffffffffffffffffffffffffffffffffffffffffffff838183161783525050505050614d86565b50505050565b7fffffffffffffffffffffffffffffffffffffffff0000000000000000000000000160001b8160000160006001600001b8152602001908152602001600020819055506001600001b8160010160007fffffffffffffffffffffffffffffffffffffffffffffffffffffffff0000000000000000000000000160001b8152602001908152602001600020819055505056 5b6000816bffffffffffffffffffffffff166060846bffffffffffffffffffffffff16901b60c08667ffffffffffffffff16901b010160001b90509392505050565b600061395d8383604051806040016040528060 1e81526020017f536166654d6174683a207375627472616374696f6e206f766572666c6f770000815250614e75565b905092915050565b60006801000000000000000082106139c8576040517f08c379a0000000000000000000000000000000000000000000000000000000008152600401808060200182810382526026815260200180615 9fd60269139604001915050604051809103 90fd5b819050919050565b60008073ffffffffffffffffffffffffffffffffffffffff168273ffffffffffffffffffffffffffffffffffffffff161415613a74576040517f08c379a000000000000000000000000000000000000000000000000000000000815260040180806020018281038252601a8152602001807f43616e6e6f7420696e73657274207a65726f20616464726573730000000000000815250602001915050604051809103 90fd5b7fffffffffffffffffffffffffffffffffffffffffffffffffffffff67ffffffffffffffff168367ffffffffffffffff161415613b1e576040517f08c379a000000000000000000000000000000000000000000000000000000000815260040180806020018281038252601881526020018 07f43616e6e6f7420696e73657274206d61782075696e7436340000000000000000815250602001915050604051809103 90fd5b60008460010160008473ffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff16815260200190815260200160002060009054906101000a900467ffffffffffffffff1667ffffffffffffffff16141580613c065750600073ffffffffffffffffffffffffffffffffffffffffffff168460000160006001860167ffffffffffffffff1667ffffffffffffffff16815260200190815260200160002060009054906101000a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff1614155b

15613c145760009050613cef565b818460000160006001860167ffffffffffffffff1667ffffffffffffffff1681
526020019081526020016000206000610100 0a81548173ffffffffffffffffffffffffffffffffffffffff0219169083
73ffffffffffffffffffffffffffffffffffffff160217905550600183018460010160008473ffffffffffffffffffffffffffffffffffffff
f1673ffffffffffffffffffffffffffffffffffffff168152602001908152602001600020600061010 00a81548167ffff
ffffffffff021916908367ffffffffffffffff160217905550600190505b9392505050565b60008083600101
60008473ffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffff168152602001908152602001600 16
000206000905490610100 0a900467ffffffffffffffff1667ffffffffffffffff161415905092915050565b60006
13d6f8383613cf6565b613de1576040517f08c379a000000000000000000000000000000000000
000000000000000000000008152600401808060200182810382526 01b8152602001807f4d75
737420686176652041646472657373207 46f20676574204944000000000815250602001915
050604051809103 90fd5b60018360010160008473ffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffff
ffffffffffffff168152602001908152602001600020600090549061010 00a900467ffffffffffffff160390
5092915050565b600806000 60c08460001c901c925060608460001c901c91508360001c905
09193909250565b60007ffffffffffffffffffffffffffffffffffffff0000000000000000000000160001b821415
613ee6576040517f08c379a000000000000000000000000000000000000000000000000000000
000008152600401808060200182810382526022815260200180 6158ba602291396040019150
506040518091039 0fd5b60008360000160008481526020019081526020016000205490506000
0801b811415613f5c576040517f08c379a000000000000000000000000000000000000000000000000000000
000000000008152600401808060200182810382526 02a81526020018061597c602a9139
604001915050604051809103 90fd5b809150509291505056 5b60006c0100000000000000000
00000008210613fcd576040517f08c379a000000000000000000000000000000000000000000000000000000
000000000008152600401808060200182810382526026815260200180 61577860269139
604001915050604051809103 90fd5b819050919050565b60006140226103e86140146002600
089815260200190815260200160002060 0a0154856bffffffffffffffffffffffff166136c490919063ffffffff1
6565b61374a90919063ffffffff16565b905060026000868152602001908152602001600020600 9
01600c9054906101000a900460ff161561407f5761407a856140728385 6bffffffffffffffffffffffff16613
63c90919063ffffffff16565b600086614466565b6141c0565b6000806140a16002600089815260
200190815260200160002060080154613e42565b92509250506000 6140c987866bffffffffffffffffffff
fffff1661391b90919063ffffffff16565b905060006141126141038 76bffffffffffffffffffffffff166140f58589
6136c490919063ffffffff16565b61374a90919063ffffffff16565b8361363c90919063ffffffff16565b90
506000614157856bffffffffffffffffffffffff16614149866bffffffffffffffffffffffff168c6136c490919063ffffffff165
65b61374a90919063ffffffff16565b90506141658a83838b614466565b6141ba8a61419c896bfffff
fffffffffffffffff1661418e8d8b6136c490919063ffffffff16565b61374a90919063ffffffff16565b600060
0a6000905490610100 0a900467ffffffffffffffff16614466565b50505050505b5050505050565b600
033905090565b6000600160001b8214156141e6576000905061420556 5b6000801b83600001
600084815260200190815260200160002054141590505b92915050565b60008061421884846
1455b565b9050801561423e576000801b8460010160008581526020019081526020016000 20
819055505b809150509291505056 5b60008060008061425786613e42565b925092509250600
080600061426b88613e42565b9250925092506142a0846bffffffffffffffffffffffff16836bffffffffffffffffffffffff

f166136c490919063ffffffff16565b6142cf826bffffffffffffffffffffffff16876bffffffffffffffffffffffff166136c4909
19063ffffffff16565b10156142e4576001965050505050505614460565b614313846bffffffffffffffff
ffffff16836bffffffffffffffffffffffff166136c490919063ffffffff16565b614342826bffffffffffffffffffffff16876bffff
ffffffffffffffffff166136c490919063ffffffff16565b11156143575760009650505050505061446056
5b816bffffffffffffffffffffffff16856bffffffffffffffffffffffff16101561438a576001965050505050506144600
565b816bffffffffffffffffffffffff16856bffffffffffffffffffffffff1611156143bd576000965050505050506144600565b8267ffffffffffffff168667ffffffffffffff16141561442a576040517f08c379a0000000000000000
0000000000000000000000000000000000000000815260040180806020018281038252602d
815260200180615a6e602d913960400191505060405180910390fd5b8267ffffffffffffff168667ffff
ffffffffff16101561445576001965050505050505614460565b60009650505050505b929
15050565b600061447c826005614f3590919063ffffffff16565b905060008411156144e9576144e
881856002600089815260200190815260200160002060000160009054906101000a900473ffff
ffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff166146069092919063ffffffff16565b5b600083
111561455457614553818460026000898152602001908152602001600020600101600090549
06101000a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff166146069092919063ff
ffffff16565b5b5050505050565b60006145678383614cf565b614574576000905061460056 5b
600083600101600084815260200190815260200160002054905060008460000160008581526
020019081526020016000205490508085600001600084815260200190815260200160002081
905550818560010160008381526020019081526020016000208190555060008 01b856000016
000868152602001908152602001600020819055506001925050505b92915050565b6146a383
63a9059cbb60e01b848460405160240180837 3ffffffffffffffffffffffffffffffffffffffff168152602001828152
602001925050506040516020818303038152906040529 07bffffffffffffffffffffffffffffffffffffffffffffffffffffffff1
916602082018051 7bffffffffffffffffffffffffffffffffffffffffffffffffffffffff838183161783525050505061 4d86565b
505050565b600080600360008a8152602001908152602001600020600090549061010 00a900
473ffffffffffffffffffffffffffffffffffffffff169050600073ffffffffffffffffffffffffffffffffffffffff168173ffffffffffffffffffffffffffffffffffffffff
ffff16146148ad576319a05a7e60e01b7bffffffffffffffffffffffffffffffffffffffffffffffffffffffff19168173ffffffffffffffffffffffffffffffffffffffff
ffffffffffffffffff166319a05a7e858c8989604051856 3ffffffff1660e01b8152600401808573ffffffffffffffffffffffffffffffffffffffff168152602001848152602001806020018281038252848482818152602001925
080828437600081840152601f19601f8201169050808301925050509550505050506020604
0518083038186803b1580156147df57600080fd5b505afa1580156147f3573d6000803e3d6000
fd5b505050506040513d602081101561480957600080fd5b810190808051906020019092919 0
5050507bffffffffffffffffffffffffffffffffffffffffffffffffffffffff1916146148ac576040517f08c379a000000000000
0000000000000000000000000000000000000000815260040180806020018281038252
601f8152602001807f75736572206e6f7420616c6c6f77656420746f20706c616365206f726465
72008152506020019150506040518091039 0fd5b5b506000806148d0600260008c815260200
190815260200160002060040154613e42565b925092505060005b895181101561 49c8576149
23898281518110 6148f057fe5b60200260200101516bffffffffffffffffffffffff16836bffffffffffffffffffffffff166
136c490919063ffffffff16565b614965846bffffffffffffffffffffffff168c84815181106149 4157fe5b602002
60200101516bffffffffffffffffffffffff166136c490919063ffffffff16565b106149bb576040517f08c379a0

0000000000000000000000000000000000000000000000000000000081526004018080602001828103825260298152602001806158956025913960400191505060405180910390fd5b8189828151811061
4a0d57fe5b6020026020010151 6bffffffffffffffffffffffff1611614a79576040517f08c379a0000000000
0000000000000000000000000000000000000000000000000000000081526004018080602001828103825
260258152602001806158956025913960400191505060405180910390fd5b8189828151811061
4a8657fe5b6020026020010151 6bffffffffffffffffffffffff1611614b0f576040517f08c379a0000000
0000000000000000000000000000000000000000000000000000000081526004018080602001828103
8252600f8152602001807f6f7264657220746f6f20736d616c6c000000000000000000000000
0000000008152506020019150506040518091039 0fd5b614b79614b43858c8481518110614
b2257fe5b6020026020010151 8c8581518110614b3657fe5b6020026020010151 6138da565b8
98381518110614b4f57fe5b6020026020010151 600160008f8152602001908152602001600 02
06150089092919063ffffffff16565b15614c4257614bb2898281518110614b8d57fe5b602002602
00101516bffffffffffffffffffffffff168461363c90919063ffffffff16565b92508367ffffffffffffff168b7f9304f2f
c7ed6d42c0400e678dbc7283e1e9054889c3afea5f965adff66ef9eac8c8481518110614bed57f
e5b6020026020010151 8c8581518110614c0157fe5b602002602001015160405180836bfffffffff
fffffffffffff168152602001826bffffffffffffffffffffffff168152602001925050506040518091039 0a35b808
06001019150506149f6565b50614cb333084600260008f8152602001908152602001600020 6
00101600090549061010 00a900473ffffffffffffffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffffff1661
37949093929190 63ffffffff16565b505097965050505050505050565b60008083118290614d6c576
040517f08c379a0000000000000000000000000000000000000000000000000000000008152
60040180806020018281038252838181518152602001915080519060200190808383600005b8
3811015614d31578082015181840152602081019050614d16565b505050509050908101906 0
1f168015614d5e57808203805160018360200361010 00a031916815260200191505b5092505
050604051809 0fd5b506000838581614d7857fe5b049050809150509392505050565b606
0614de882604051806040016040528060208152602001 7f5361666545524332303a206c6f772
d6c6576656c2063616c6c206661696c65648152508573ffffffffffffffffffffffffffffffffffffffff166152b69092
919063ffffffff16565b905060008151111 5614e705780806020019051602081101 5614e0957600
080fd5b810190808051906020019092919050505050614e6f576040517f08c379a000000000000
0000000000000000000000000000000000000000000000000000000081526004018080602001828103825 2
602a815260200180615aed602a91396040019150506040518091039 0fd5b5b505050565b600
0838311158290614f22576040517f08c379a0000000000000000000000000000000000000000000
0000000000000000081526004018080602001828103825283818151815260200191508051906
0200190808383600005b83811015614ee75780820151818401526020810190506 14ecc565b5
0505050905090810190601f168015614f1457808203805160018360200361010 00a031916815
260200191505b5092505050604051809 0fd5b506000838503905080915050939250505050
565b6000614f4183836152ce565b614fb3576040517f08c379a0000000000000000000000000
0000000000000000000000000000000081526004018080602001828103825260 1b815260200

1807f4d75737420686176652049442074 6f20676574204164647265737300000000008152506
020019150506040518091039 0fd5b826000016000600184016 7fffffffffffffff1667fffffffffffffff16815
260200190815260200160002060009054906101000a900473fffffffffffffffffffffffffffffffffffffff1690509
2915050565b6000806150148461 3e42565b9250505060006bfffffffffffffffffffffff16816bffffffffffffffffff
fffff1614156150ab576040517f08c379a0000000000000000000000000000000000000000000000000
0000000000008152600401808060200182810382526 01f8152602001807f496e73657274696
e67207a65726f206973206e6f7420737570706f72746564008152506020019150506040518 09
10390fd5b600160001b84141580156150 e257507fffffffffffffffffffffffffffffffffffffff0000000000000000
0000000160001b8414155b615154576040517f08c379a0000000000000000000000000000000
0000000000000000000000000008152600401808060200182810382526 01e8152602001807f
496e73657274696e6720656c656d656e74206973206e6f742076616c6964000081525060200
191505060405180910390fd5b61515e85856141cf565b1561516d576000915050615 2af565b6
00160001b831415801561519857506000801b856001016000858152602001908152602001600
02054145b156151a7576000915050615 2af565b6151ba848461424890919063ffffffff16565b6
151c857600915050615 2af565b5b6000801b856000016000858152602001908152602001600
0020541415615206578460010160008481526020019081526020016000205492506151c9565
b6000808490505b809150866000016000828152602001908152602001600020549050615 23b
868261424890919063ffffffff16565b61520d57858760000160008481526020019081526020016
0002081905550858760010160008381526020019081526020016000208190555081876001016
0008881526020019081526020016000208190555080876000016000888152602001908152 6
0200160002081905550600193505050505b9392505050565b60606152c58484600085615354
565b905093925050505 65b60008073fffffffffffffffffffffffffffffffffffffff16836000016000600185016 7fffff
ffffffffff1667fffffffffffffff16815260200190815260200160002060009054906101000a900473ffffffff
ffffffffffffffffffffffffffff1673ffffffffffffffffffffffffffffffffffffff16141590509291505 0565b6060824710156153
af576040517f08c379a0000000000000000000000000000000000000000000000000000000000000
8152600401808060200182810382526026815260200180615 8dc60269139604001915050604
05180910390fd5b6153b8856154fd565b61542a576040517f08c379a0000000000000000000
0000000000000000000000000000000000008152600401808060200182810382526 01d8152
602001807f416464726573733a2063616c6c20746f206e6f6e2d636f6e7472616374000000815
2506020019150506040518091039 0fd5b600060608673fffffffffffffffffffffffffffffffffffffff168587604051
8082805190602001908083835b6020831061547a57805182526020820191506020810190506
0208303925061545756 5b60018360200361010 00a038019825116818451168082178552505 0
5050505090500191505060006040518083038185875af1925050503d80600081146154dc576
040519150601f19603f3d011682016040523d82523d6000602084013e6154e1565b606091505
b5091509150615 4f1828286615510565b9250505094935050505 0565b600080823b90506000
811191505091905 0565b606083156155205782905061 55d5565b600083511115615533578 25
18084602001fd5b816040517f08c379a0000000000000000000000000000000000000000000000000
0000000000008152600401808060200182810382528381815181526020019150805190602 0
190808383600 5b8381101561559a57808201518184015260208101905061557f565b50505

0509050908101906 01f1680156155c75780820380516001836020036101000a031916815260
200191505b5092505050604051809103 90fd5b9392505050565b828054600181600116156101
00020316600290049060005260206000209060 1f016020900481019282601f1061561d57805
160ff191683800117855561564b565b828001600101855582156 1564b579182015b828111156
1564a57825182559160200191906001019061562f565b5b5090506156589190615 75a565b50
90565b508054600181600116156101000203166002900460008255806 01f106156825750615
6a1565b601f016020900490600052602060002090810190615 6a0919061575a565b5b50565b
604051806101c00160405280600073ffffffffffffffffffffffffffffffffffffffff168152602001600073ffffffffffffffffff
ffffffffffffffffffff168152602001600081526020016000815260200160008019168152602001600081526020016000815260
200160008152602001600080191681526020016000801916815260200160006bffffffffff
ffffffffffff168152602001600015158152602001600015158152602001600081526020016000815
25090565b5b808211156157357600081600090555060010161575b565b509056fe5361666
5436173743a2076616c756520646f65736e27742066697420696e2039362026974736e6f742
0616c6c6f77656420746f20736574746c652061756374696f6e2061746f6d6963616c6c797072
6563616c63756c61746553656c6c416d6f756e7453756d20697320616c726561647920746f6f2
0616476616e6365644f6e6c792074686520757365722063616e2063616e63656c2068697320
6f72646572734f776e61626c653a206e6577206f776e65722069732074686520 7a65726f20616
46472657373746f6b656e732063616e6e6f742062652061756374696f6e656420666f72206672
65656e6f206c6f6e67657220696e206f7264657220706c6163656d656e7420616e642063616e
63656c6174696f6e2070686173655f6d696e427579416d6f756e7473206d7573742062652067
726561746572207468616e2030547279696e6720746f2067657420 6e657874206f66206c6173
7420656c656d656e74416464726573733a20696e737566666969636 9656e742062616c616e636
520666f722063616c6c466565206973206e6f7420616c6c6f77656420746f2062652073657420
686967686572207468616e20312e352561756374696f20656e642064617465206d7573742
0626520696e207468652066757475726 54f6e6c79206f6e65206f726465722063616e2062652
0706c616365642061746f6d6963616c6c79547279696e6720746f20676574206e657874206f6
6206e6f6e2d6578697374656e7420656c656d656e746d696e696d756d42696464696e67416d
6f756e745065724f726465722069732 06e6f7420616c6c6f77656420746f206265207a65726f53
6166654d6174683a206d756c7469706c69636174696f6e206f766572666c6f77536166654361
73743a2076616c756520646f65736e27742066697420696e203634206974736f6e6c792061
6c6c6f77656420746f20636c61696d20666f722073616d652075736572417 56374696f6e206e6
f7420696e20736f6c7574696f6e207375626d697373696f6e207068617365757365722069732
06e6f7420616c6c6f77656420746f20706c6163652073616d65206f726465722074776963656c6
96d69742070726963 65206e6f74206265747465722074686 16e206d696d696d616c206f66666
57274696d6520706572696f647320617265206e6f7420636f6e6669677572656420636f727265
63746c795361666545524332303a204552433230206f706572617 4696f6e20646964206e6f74
207375636365646e6f206c6f6e67657220696e206f7264657220706c6163656d656e742070
68617365a2646970667358221 2209d3f3585da1ffe9756c8a2e5f28991662e7f098068faf72dd2
ec014b6bd536c964736f6c634300060c0033", "devdoc": { "kind": "dev", "methods": { "owner()":

{ "details": "Returns the address of the current owner." }, "renounceOwnership()": { "details": "Leaves the contract without owner. It will not be possible to call `onlyOwner` functions anymore. Can only be called by the current owner. NOTE: Renouncing ownership will leave the contract without an owner, thereby removing any functionality that is only available to the owner." }, "transferOwnership(address)": { "details": "Transfers ownership of the contract to a new account (`newOwner`). Can only be called by the current owner." } }, "version": 1 }, "userdoc": { "kind": "user", "methods": {}, "version": 1 }, "storageLayout": { "storage": [{ "astId": 30, "contract": "contracts/EasyAuction.sol:EasyAuction", "label": "_owner", "offset": 0, "slot": "0", "type": "t_address" }, { "astId": 1172, "contract": "contracts/EasyAuction.sol:EasyAuction", "label": "sellOrders", "offset": 0, "slot": "1", "type": "t_mapping(t_uint256,t_struct(Data)3111_storage)" }, { "astId": 1176, "contract": "contracts/EasyAuction.sol:EasyAuction", "label": "auctionData", "offset": 0, "slot": "2", "type": "t_mapping(t_uint256,t_struct(AuctionData)1168_storage)" }, { "astId": 1180, "contract": "contracts/EasyAuction.sol:EasyAuction", "label": "auctionAccessManager", "offset": 0, "slot": "3", "type": "t_mapping(t_uint256,t_address)" }, { "astId": 1184, "contract": "contracts/EasyAuction.sol:EasyAuction", "label": "auctionAccessData", "offset": 0, "slot": "4", "type": "t_mapping(t_uint256,t_bytes_storage)" }, { "astId": 1186, "contract": "contracts/EasyAuction.sol:EasyAuction", "label": "registeredUsers", "offset": 0, "slot": "5", "type": "t_struct(Data)2920_storage" }, { "astId": 1188, "contract": "contracts/EasyAuction.sol:EasyAuction", "label": "numUsers", "offset": 0, "slot": "7", "type": "t_uint64" }, { "astId": 1190, "contract": "contracts/EasyAuction.sol:EasyAuction", "label": "auctionCounter", "offset": 0, "slot": "8", "type": "t_uint256" }, { "astId": 1199, "contract": "contracts/EasyAuction.sol:EasyAuction", "label": "feeNumerator", "offset": 0, "slot": "9", "type": "t_uint256" }, { "astId": 1205, "contract": "contracts/EasyAuction.sol:EasyAuction", "label": "feeReceiverUserId", "offset": 0, "slot": "10", "type": "t_uint64" }], "types": { "t_address": { "encoding": "inplace", "label": "address", "numberOfBytes": "20" }, "t_bool": { "encoding": "inplace", "label": "bool", "numberOfBytes": "1" }, "t_bytes32": { "encoding": "inplace", "label": "bytes32", "numberOfBytes": "32" }, "t_bytes_storage": { "encoding": "bytes", "label": "bytes", "numberOfBytes": "32" }, "t_contract(IERC20)478": { "encoding": "inplace", "label": "contract IERC20", "numberOfBytes": "20" }, "t_mapping(t_address,t_uint64)": { "encoding": "mapping", "key": "t_address", "label": "mapping(address => uint64)", "numberOfBytes": "32", "value": "t_uint64" }, "t_mapping(t_bytes32,t_bytes32)": { "encoding": "mapping", "key": "t_bytes32", "label": "mapping(bytes32 => bytes32)", "numberOfBytes": "32", "value": "t_bytes32" }, "t_mapping(t_uint256,t_address)": { "encoding": "mapping", "key": "t_uint256", "label": "mapping(uint256 => address)", "numberOfBytes": "32", "value": "t_address" }, "t_mapping(t_uint256,t_bytes_storage)": { "encoding": "mapping", "key": "t_uint256", "label": "mapping(uint256 => bytes)", "numberOfBytes": "32", "value": "t_bytes_storage" }, "t_mapping(t_uint256,t_struct(AuctionData)1168_storage)": { "encoding": "mapping", "key": "t_uint256", "label": "mapping(uint256 => struct EasyAuction.AuctionData)", "numberOfBytes":

"32", "value": "t_struct(AuctionData)1168_storage" },
"t_mapping(t_uint256,t_struct(Data)3111_storage)": { "encoding": "mapping", "key":
"t_uint256", "label": "mapping(uint256 => struct IterableOrderedOrderSet.Data)",
"numberOfBytes": "32", "value": "t_struct(Data)3111_storage" },
"t_mapping(t_uint64,t_address)": { "encoding": "mapping", "key": "t_uint64", "label":
"mapping(uint64 => address)", "numberOfBytes": "32", "value": "t_address" },
"t_struct(AuctionData)1168_storage": { "encoding": "inplace", "label": "struct
EasyAuction.AuctionData", "members": [{ "astId": 1141, "contract":
"contracts/EasyAuction.sol:EasyAuction", "label": "auctioningToken", "offset": 0, "slot": "0",
"type": "t_contract(IERC20)478" }, { "astId": 1143, "contract":
"contracts/EasyAuction.sol:EasyAuction", "label": "biddingToken", "offset": 0, "slot": "1", "type":
"t_contract(IERC20)478" }, { "astId": 1145, "contract":
"contracts/EasyAuction.sol:EasyAuction", "label": "orderCancellationEndDate", "offset": 0,
"slot": "2", "type": "t_uint256" }, { "astId": 1147, "contract":
"contracts/EasyAuction.sol:EasyAuction", "label": "auctionEndDate", "offset": 0, "slot": "3",
"type": "t_uint256" }, { "astId": 1149, "contract": "contracts/EasyAuction.sol:EasyAuction",
"label": "initialAuctionOrder", "offset": 0, "slot": "4", "type": "t_bytes32" }, { "astId": 1151,
"contract": "contracts/EasyAuction.sol:EasyAuction", "label":
"minimumBiddingAmountPerOrder", "offset": 0, "slot": "5", "type": "t_uint256" }, { "astId": 1153,
"contract": "contracts/EasyAuction.sol:EasyAuction", "label": "interimSumBidAmount", "offset":
0, "slot": "6", "type": "t_uint256" }, { "astId": 1155, "contract":
"contracts/EasyAuction.sol:EasyAuction", "label": "interimOrder", "offset": 0, "slot": "7", "type":
"t_bytes32" }, { "astId": 1157, "contract": "contracts/EasyAuction.sol:EasyAuction", "label":
"clearingPriceOrder", "offset": 0, "slot": "8", "type": "t_bytes32" }, { "astId": 1159, "contract":
"contracts/EasyAuction.sol:EasyAuction", "label": "volumeClearingPriceOrder", "offset": 0,
"slot": "9", "type": "t_uint96" }, { "astId": 1161, "contract":
"contracts/EasyAuction.sol:EasyAuction", "label": "minFundingThresholdNotReached",
"offset": 12, "slot": "9", "type": "t_bool" }, { "astId": 1163, "contract":
"contracts/EasyAuction.sol:EasyAuction", "label": "isAtomicClosureAllowed", "offset": 13, "slot":
"9", "type": "t_bool" }, { "astId": 1165, "contract": "contracts/EasyAuction.sol:EasyAuction",
"label": "feeNumerator", "offset": 0, "slot": "10", "type": "t_uint256" }, { "astId": 1167, "contract":
"contracts/EasyAuction.sol:EasyAuction", "label": "minFundingThreshold", "offset": 0, "slot":
"11", "type": "t_uint256" }], "numberOfBytes": "384" }, "t_struct(Data)2920_storage": {
"encoding": "inplace", "label": "struct IdToAddressBiMap.Data", "members": [{ "astId": 2915,
"contract": "contracts/EasyAuction.sol:EasyAuction", "label": "idToAddress", "offset": 0, "slot":
"0", "type": "t_mapping(t_uint64,t_address)" }, { "astId": 2919, "contract":
"contracts/EasyAuction.sol:EasyAuction", "label": "addressToId", "offset": 0, "slot": "1", "type":
"t_mapping(t_address,t_uint64)" }], "numberOfBytes": "64" }, "t_struct(Data)3111_storage": {
"encoding": "inplace", "label": "struct IterableOrderedOrderSet.Data", "members": [{ "astId":

3106, "contract": "contracts/EasyAuction.sol:EasyAuction", "label": "nextMap", "offset": 0, "slot": "0", "type": "t_mapping(t_bytes32,t_bytes32)" }, { "astId": 3110, "contract": "contracts/EasyAuction.sol:EasyAuction", "label": "prevMap", "offset": 0, "slot": "1", "type": "t_mapping(t_bytes32,t_bytes32)" }], "numberOfBytes": "64" }, "t_uint256": { "encoding": "inplace", "label": "uint256", "numberOfBytes": "32" }, "t_uint64": { "encoding": "inplace", "label": "uint64", "numberOfBytes": "8" }, "t_uint96": { "encoding": "inplace", "label": "uint96", "numberOfBytes": "12" } } } }