// SPDX-License-Identifier: MIT /\* MIT License Copyright (c) 2018 requestnetwork Copyright (c) 2018 Fragments, Inc. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE. \*/ pragma solidity ^0.6.2; /\*\* \* @title SafeMathInt \* @dev Math operations for int256 with overflow safety checks. \*/ library SafeMathInt { int256 private constant MIN\_INT256 = int256(1) << 255; int256 private constant MAX\_INT256 = ~(int256(1) << 255); /\*\* \* @dev Multiplies two int256 variables and fails on overflow. \*/ function mul(int256 a, int256 b) internal pure returns (int256) { int256 c = a \* b; // Detect overflow when multiplying MIN\_INT256 with -1 require(c != MIN\_INT256 || (a & MIN\_INT256) != (b & MIN\_INT256)); require((b == 0) || (c / b == a)); return c; } /\*\* \* @dev Division of two int256 variables and fails on overflow. \*/ function div(int256 a, int256 b) internal pure returns (int256) { // Prevent overflow when dividing MIN\_INT256 by -1 require(b != -1 || a != MIN\_INT256); // Solidity already throws when dividing by 0. return a / b; } /\*\* \* @dev Subtracts two int256 variables and fails on overflow. \*/ function sub(int256 a, int256 b) internal pure returns (int256) { int256 c = a - b; require((b >= 0 && c <= a) || (b < 0 && c > a)); return c; } /\*\* \* @dev Adds two int256 variables and fails on overflow. \*/ function add(int256 a, int256 b) internal pure returns (int256) { int256 c = a + b; require((b >= 0 && c >= a) || (b < 0 && c < a)); return c; } /\*\* \* @dev Converts to absolute value, and fails on overflow. \*/ function abs(int256 a) internal pure returns (int256) { require(a != MIN\_INT256); return a < 0 ? -a : a; } function toUint256Safe(int256 a) internal pure returns (uint256) { require(a >= 0); return uint256(a); } }