**Library**

Libraries are similar to contracts, but you can't declare any state variable and you can't send ether.

A library is embedded into the contract if all library functions are internal.

Otherwise the library must be deployed and then linked before the contract is deployed.

*// SPDX-License-Identifier: MIT*

pragma solidity ^0.8.13;

library SafeMath {

function add(uint x, uint y) internal pure returns (uint) {

uint z = x + y;

require(z >= x, "uint overflow");

return z;

}

}

library Math {

function sqrt(uint y) internal pure returns (uint z) {

if (y > 3) {

z = y;

uint x = y / 2 + 1;

while (x < z) {

z = x;

x = (y / x + x) / 2;

}

} else if (y != 0) {

z = 1;

}

*// else z = 0 (default value)*

}

}

contract TestSafeMath {

using SafeMath for uint;

uint public MAX\_UINT = 2\*\*256 - 1;

function testAdd(uint x, uint y) public pure returns (uint) {

return x.add(y);

}

function testSquareRoot(uint x) public pure returns (uint) {

return Math.sqrt(x);

}

}

*// Array function to delete element at index and re-organize the array*

*// so that their are no gaps between the elements.*

library Array {

function remove(uint[] storage arr, uint index) public {

*// Move the last element into the place to delete*

require(arr.length > 0, "Can't remove from empty array");

arr[index] = arr[arr.length - 1];

arr.pop();

}

}

contract TestArray {

using Array for uint[];

uint[] public arr;

function testArrayRemove() public {

for (uint i = 0; i < 3; i++) {

arr.push(i);

}

arr.remove(1);

assert(arr.length == 2);

assert(arr[0] == 0);

assert(arr[1] == 2);

}

}