pragma solidity ^0.8.0;

contract LandRegistry {

struct Land {

uint256 id;

string location;

uint256 area;

address owner;

bool registered;

}

mapping(uint256 => Land) public lands;

event LandRegistered(uint256 indexed landId, string location, uint256 area, address indexed owner);

event OwnershipTransferred(uint256 indexed landId, address indexed oldOwner, address indexed newOwner);

function registerLand(uint256 \_id, string memory \_location, uint256 \_area) public {

require(!lands[\_id].registered, "Land already registered");

lands[\_id] = Land({

id: \_id,

location: \_location,

area: \_area,

owner: msg.sender,

registered: true

});

emit LandRegistered(\_id, \_location, \_area, msg.sender);

}

function transferOwnership(uint256 \_id, address \_newOwner) public {

Land storage land = lands[\_id];

require(land.registered, "Land not registered");

require(land.owner == msg.sender, "Only owner can transfer ownership");

require(\_newOwner != address(0), "Invalid address");

address oldOwner = land.owner;

land.owner = \_newOwner;

emit OwnershipTransferred(\_id, oldOwner, \_newOwner);

}

function getLand(uint256 \_id) public view returns (

uint256 id,

string memory location,

uint256 area,

address owner,

bool registered

) {

Land memory land = lands[\_id];

require(land.registered, "Land not registered");

return (land.id, land.location, land.area, land.owner, land.registered);

}

function isLandRegistered(uint256 \_id) public view returns (bool) {

return lands[\_id].registered;

}

}