# Solidity Contract JSON

{  
 "language": "Solidity",  
 "sources": {  
 "AujCoin.sol": {  
 "content": "// SPDX-License-Identifier: MIT  
pragma solidity ^0.8.20;  
  
contract AujCoin {  
 string public name = "AujCoin";  
 string public symbol = "auj";  
 uint8 public decimals = 18;  
 uint256 public totalSupply;  
  
 address public owner;  
 mapping(address => uint256) public balanceOf;  
 mapping(address => mapping(address, uint256)) public allowance;  
  
 event Transfer(address indexed from, address indexed to, uint256 value);  
 event Approval(address indexed owner, address indexed spender, uint256 value);  
 event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);  
  
 constructor(uint256 \_initialSupply) {  
 owner = msg.sender;  
 totalSupply = \_initialSupply \* 10 \*\* uint256(decimals);  
 balanceOf[owner] = totalSupply;  
 emit Transfer(address(0), owner, totalSupply);  
 }  
  
 modifier onlyOwner() {  
 require(msg.sender == owner, "Ownable: caller is not the owner");  
 \_;  
 }  
  
 function transfer(address to, uint256 value) public returns (bool) {  
 require(to != address(0), "ERC20: transfer to zero address");  
 require(balanceOf[msg.sender] >= value, "ERC20: transfer exceeds balance");  
 balanceOf[msg.sender] -= value;  
 balanceOf[to] += value;  
 emit Transfer(msg.sender, to, value);  
 return true;  
 }  
  
 function approve(address spender, uint256 value) public returns (bool) {  
 require(spender != address(0), "ERC20: approve to zero address");  
 allowance[msg.sender][spender] = value;  
 emit Approval(msg.sender, spender, value);  
 return true;  
 }  
  
 function transferFrom(address from, address to, uint256 value) public returns (bool) {  
 require(to != address(0), "ERC20: transfer to zero address");  
 require(balanceOf[from] >= value, "ERC20: transfer exceeds balance");  
 require(allowance[from][msg.sender] >= value, "ERC20: transfer exceeds allowance");  
 balanceOf[from] -= value;  
 balanceOf[to] += value;  
 allowance[from][msg.sender] -= value;  
 emit Transfer(from, to, value);  
 return true;  
 }  
  
 function increaseAllowance(address spender, uint256 addedValue) public returns (bool) {  
 require(spender != address(0), "ERC20: approve to zero address");  
 allowance[msg.sender][spender] += addedValue;  
 emit Approval(msg.sender, spender, allowance[msg.sender][spender]);  
 return true;  
 }  
  
 function decreaseAllowance(address spender, uint256 subtractedValue) public returns (bool) {  
 require(spender != address(0), "ERC20: approve to zero address");  
 require(allowance[msg.sender][spender] >= subtractedValue, "ERC20: decreased allowance below zero");  
 allowance[msg.sender][spender] -= subtractedValue;  
 emit Approval(msg.sender, spender, allowance[msg.sender][spender]);  
 return true;  
 }  
  
 function transferOwnership(address newOwner) public onlyOwner {  
 require(newOwner != address(0), "Ownable: new owner zero address");  
 emit OwnershipTransferred(owner, newOwner);  
 owner = newOwner;  
 }  
  
 function balanceOfAccount(address account) public view returns (uint256) {  
 return balanceOf[account];  
 }  
  
 function allowanceOf(address ownerAddress, address spenderAddress) public view returns (uint256) {  
 return allowance[ownerAddress][spenderAddress];  
 }  
}"  
 }  
 },  
 "settings": {  
 "optimizer": {  
 "enabled": true,  
 "runs": 200  
 },  
 "evmVersion": "london",  
 "metadata": {  
 "bytecodeHash": "ipfs"  
 },  
 "compilationTarget": {  
 "AujCoin.sol": "AujCoin"  
 }  
 }  
}