1. 荷兰拍卖

// SPDX-License-Identifier: MIT

pragma solidity ^ 0.8.17;

//荷兰拍卖合约：荷兰拍卖是指随着时间的流逝 郁金香的成色变差 因此价格贬低，所以第一个举牌人就是竞拍成功的人

//调用ERC721 中交易方法

//必须部署erc721合约 并且将拍卖地址授权

interface IERC721{

  function transferfrom(address from,address to,uint256 tokenId)external;

}

contract DutchAuction{

    event Log(uint refund);

    uint private constant DURATION=7 days;//周期

    IERC721 public  nft;

    uint public nftid;

    address public seller;//售卖方地址

    uint public startingprice;//竞拍价格

    uint public startat;//开始时间

    uint public expiresat;//结束时间

    uint public discountRate;//贬值率

    constructor(uint \_startingPrice,uint \_discountRate,address \_nft,uint \_nftid){

        seller=payable(msg.sender);

        startingprice=\_startingPrice;

        discountRate=\_discountRate;

        startat=block.timestamp;

        expiresat=block.timestamp+DURATION;

        require(\_startingPrice>=\_discountRate\*DURATION,"starting price<discount");//判断起拍价必须大于结束价格

        nft=IERC721(\_nft);

        nftid=\_nftid;

    }

    function getprice()public view returns(uint){

        uint timeelapsed=block.timestamp-startat;

        uint discount=timeelapsed\*discountRate;

        return startingprice-discount;

    }

    function buy()external payable{

        require(block.timestamp<expiresat,"auction expired");

        uint price=getprice();

        require(msg.value>=price,"eth<price");

        nft.transferfrom(seller,msg.sender,nftid);

        uint refund=msg.value-price;

        if(refund>0){

            payable(msg.sender).transfer(refund);

        }

        emit Log(refund);

        selfdestruct(payable(seller));//摧毁合约 一个合约竞拍1个nft

    }

}

1. 英国拍卖

// SPDX-License-Identifier: MIT

pragma solidity ^ 0.8.17;

interface IERC721{

  function transferfrom(address from,address to,uint256 tokenId)external;

}

contract EnglishAuction{

    event Start();

    event Bid(address indexed sender,uint amount);

    event Withdraw(address indexed bidder,uint amount);

    event End(address highestBidder,uint highestBid);

    IERC721 public nft;

    uint public nftid;

    address payable public   seller;

    uint32 public endAt;

    bool public started;

    bool public ended;

    address public highestBidder;

    uint public highestBid;

    mapping(address=>uint)public bids;

constructor(address \_nft,uint \_nftid,uint \_startingBid){

    nft=IERC721(\_nft);

    nftid=\_nftid;

    highestBid=\_startingBid;

    seller=payable( msg.sender);

}

function start()external{

    require(!started,"started");

    require(msg.sender==seller,"not seller");

    started=true;

    endAt=uint32(block.timestamp+300);

    nft.transferfrom(seller,address(this),nftid);

    emit Start();

}

function bid()external payable{

    require(started,"not started");

    require(block.timestamp<endAt,"ended");

    require(msg.value>highestBid,"value<hightest bid");

    if(highestBidder!=address(0)){

        bids[highestBidder]=highestBid+bids[highestBidder];

    }

    highestBid=msg.value;

    highestBidder=msg.sender;

    emit Bid(msg.sender,msg.value);

}

function withdraw()external{

    uint bal=bids[msg.sender];

    bids[msg.sender]=0;

    payable(msg.sender).transfer(bal);

    emit Withdraw(msg.sender,bal);

}

function end()external{

      require(started,"not started");

      require(!ended,"ended");

      require(block.timestamp>=endAt,"not ended");

      ended=true;

      if(highestBidder!=address(0)){

          nft.transferfrom(address(this),highestBidder,nftid);

         seller.transfer(highestBid);

      }else{

          nft.transferfrom(address(this),seller,nftid);

      }

      emit End(highestBidder,highestBid);

}

}