In this notebook, I will be attempting to figure out ways to check how similar two protocols are to each other. For the sake of the example, I am taking two protocols:

Pancakeswap on BSC UniswapV2 on Etereum

I already know that pancakeswap is a fork of uniswap.

The two approaches used to do the task are :-

1. Compare the codes line by line by using jaro distance metric between 2 strings. (https://rosettacode.org/wiki/Jaro_similarity (https://rosettacode.org/wiki/Jaro_similarity))
2. Check the functions of both the protocols (smart contracts) to see if they have the same functions.

For the sake of this notebook, I copied the exact contract code from etherscan but if we agree to proceed in this direction, I will try to figure out a way to get that programatically.

Imports and general dependencies

In [1]:
```
from web3 import Web3
import jellyfish
infura_url = "https://mainnet.infura.io/v3/cab1dcbaa74a458fb227f720b6f25cf5
w3 = Web3(Web3.HTTPProvider(infura_url))
w3.isConnected()
```

Out[1]:
```
True
```

constants

In [2]:
```
ype":"uint256[]"}],"stateMutability":"nonpayable","type":"function"},{"state

ype":"uint256[]"}],"stateMutability":"nonpayable","type":"function"},{"state
```

In [3]:
```
uniswapV2RouterCode = """
/**
 *Submitted for verification at Etherscan.io on 2020-06-05
*/

pragma solidity =0.6.6;

interface IUniswapV2Factory {
    event PairCreated(address indexed token0, address indexed token1, addre

    function feeTo() external view returns (address);
    function feeToSetter() external view returns (address);

    function getPair(address tokenA, address tokenB) external view returns
    function allPairs(uint) external view returns (address pair);
    function allPairsLength() external view returns (uint);

    function createPair(address tokenA, address tokenB) external returns (a

    function setFeeTo(address) external;
    function setFeeToSetter(address) external;
}

interface IUniswapV2Pair {
    event Approval(address indexed owner, address indexed spender, uint val
    event Transfer(address indexed from, address indexed to, uint value);

    function name() external pure returns (string memory);
    function symbol() external pure returns (string memory);
    function decimals() external pure returns (uint8);
    function totalSupply() external view returns (uint);
    function balanceOf(address owner) external view returns (uint);
    function allowance(address owner, address spender) external view return

    function approve(address spender, uint value) external returns (bool);
    function transfer(address to, uint value) external returns (bool);
    function transferFrom(address from, address to, uint value) external re

    function DOMAIN_SEPARATOR() external view returns (bytes32);
    function PERMIT_TYPEHASH() external pure returns (bytes32);
    function nonces(address owner) external view returns (uint);

    function permit(address owner, address spender, uint value, uint deadli

    event Mint(address indexed sender, uint amount0, uint amount1);
    event Burn(address indexed sender, uint amount0, uint amount1, address
    event Swap(
        address indexed sender,
        uint amount0In,
        uint amount1In,
        uint amount0Out,
        uint amount1Out,
        address indexed to
    );
    event Sync(uint112 reserve0, uint112 reserve1);
```

```
    function MINIMUM_LIQUIDITY() external pure returns (uint);
    function factory() external view returns (address);
    function token0() external view returns (address);
    function token1() external view returns (address);
    function getReserves() external view returns (uint112 reserve0, uint112
    function price0CumulativeLast() external view returns (uint);
    function price1CumulativeLast() external view returns (uint);
    function kLast() external view returns (uint);

    function mint(address to) external returns (uint liquidity);
    function burn(address to) external returns (uint amount0, uint amount1)
    function swap(uint amount0Out, uint amount1Out, address to, bytes calld
    function skim(address to) external;
    function sync() external;

    function initialize(address, address) external;
}

interface IUniswapV2Router01 {
    function factory() external pure returns (address);
    function WETH() external pure returns (address);

    function addLiquidity(
        address tokenA,
        address tokenB,
        uint amountADesired,
        uint amountBDesired,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline
    ) external returns (uint amountA, uint amountB, uint liquidity);
    function addLiquidityETH(
        address token,
        uint amountTokenDesired,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) external payable returns (uint amountToken, uint amountETH, uint liqu
    function removeLiquidity(
        address tokenA,
        address tokenB,
        uint liquidity,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline
    ) external returns (uint amountA, uint amountB);
    function removeLiquidityETH(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) external returns (uint amountToken, uint amountETH);
```

```
    function removeLiquidityWithPermit(
        address tokenA,
        address tokenB,
        uint liquidity,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline,
        bool approveMax, uint8 v, bytes32 r, bytes32 s
    ) external returns (uint amountA, uint amountB);
    function removeLiquidityETHWithPermit(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline,
        bool approveMax, uint8 v, bytes32 r, bytes32 s
    ) external returns (uint amountToken, uint amountETH);
    function swapExactTokensForTokens(
        uint amountIn,
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    ) external returns (uint[] memory amounts);
    function swapTokensForExactTokens(
        uint amountOut,
        uint amountInMax,
        address[] calldata path,
        address to,
        uint deadline
    ) external returns (uint[] memory amounts);
    function swapExactETHForTokens(uint amountOutMin, address[] calldata pa
        external
        payable
        returns (uint[] memory amounts);
    function swapTokensForExactETH(uint amountOut, uint amountInMax, addres
        external
        returns (uint[] memory amounts);
    function swapExactTokensForETH(uint amountIn, uint amountOutMin, addres
        external
        returns (uint[] memory amounts);
    function swapETHForExactTokens(uint amountOut, address[] calldata path,
        external
        payable
        returns (uint[] memory amounts);

    function quote(uint amountA, uint reserveA, uint reserveB) external pur
    function getAmountOut(uint amountIn, uint reserveIn, uint reserveOut) e
    function getAmountIn(uint amountOut, uint reserveIn, uint reserveOut) e
    function getAmountsOut(uint amountIn, address[] calldata path) external
    function getAmountsIn(uint amountOut, address[] calldata path) external
}

interface IUniswapV2Router02 is IUniswapV2Router01 {
    function removeLiquidityETHSupportingFeeOnTransferTokens(
```

```
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) external returns (uint amountETH);
    function removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline,
        bool approveMax, uint8 v, bytes32 r, bytes32 s
    ) external returns (uint amountETH);

    function swapExactTokensForTokensSupportingFeeOnTransferTokens(
        uint amountIn,
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    ) external;
    function swapExactETHForTokensSupportingFeeOnTransferTokens(
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    ) external payable;
    function swapExactTokensForETHSupportingFeeOnTransferTokens(
        uint amountIn,
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    ) external;
}

interface IERC20 {
    event Approval(address indexed owner, address indexed spender, uint val
    event Transfer(address indexed from, address indexed to, uint value);

    function name() external view returns (string memory);
    function symbol() external view returns (string memory);
    function decimals() external view returns (uint8);
    function totalSupply() external view returns (uint);
    function balanceOf(address owner) external view returns (uint);
    function allowance(address owner, address spender) external view return

    function approve(address spender, uint value) external returns (bool);
    function transfer(address to, uint value) external returns (bool);
    function transferFrom(address from, address to, uint value) external re
}

interface IWETH {
    function deposit() external payable;
```

```
    function transfer(address to, uint value) external returns (bool);
    function withdraw(uint) external;
}

contract UniswapV2Router02 is IUniswapV2Router02 {
    using SafeMath for uint;

    address public immutable override factory;
    address public immutable override WETH;

    modifier ensure(uint deadline) {
        require(deadline >= block.timestamp, 'UniswapV2Router: EXPIRED');
        _;
    }

    constructor(address _factory, address _WETH) public {
        factory = _factory;
        WETH = _WETH;
    }

    receive() external payable {
        assert(msg.sender == WETH); // only accept ETH via fallback from th
    }

    // **** ADD LIQUIDITY ****
    function _addLiquidity(
        address tokenA,
        address tokenB,
        uint amountADesired,
        uint amountBDesired,
        uint amountAMin,
        uint amountBMin
    ) internal virtual returns (uint amountA, uint amountB) {
        // create the pair if it doesn't exist yet
        if (IUniswapV2Factory(factory).getPair(tokenA, tokenB) == address(0
            IUniswapV2Factory(factory).createPair(tokenA, tokenB);
        }
        (uint reserveA, uint reserveB) = UniswapV2Library.getReserves(facto
        if (reserveA == 0 && reserveB == 0) {
            (amountA, amountB) = (amountADesired, amountBDesired);
        } else {
            uint amountBOptimal = UniswapV2Library.quote(amountADesired, re
            if (amountBOptimal <= amountBDesired) {
                require(amountBOptimal >= amountBMin, 'UniswapV2Router: INS
                (amountA, amountB) = (amountADesired, amountBOptimal);
            } else {
                uint amountAOptimal = UniswapV2Library.quote(amountBDesired
                assert(amountAOptimal <= amountADesired);
                require(amountAOptimal >= amountAMin, 'UniswapV2Router: INS
                (amountA, amountB) = (amountAOptimal, amountBDesired);
            }
        }
    }
    function addLiquidity(
        address tokenA,
        address tokenB,
        uint amountADesired,
```

```
        uint amountBDesired,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline
    ) external virtual override ensure(deadline) returns (uint amountA, uin
        (amountA, amountB) = _addLiquidity(tokenA, tokenB, amountADesired,
        address pair = UniswapV2Library.pairFor(factory, tokenA, tokenB);
        TransferHelper.safeTransferFrom(tokenA, msg.sender, pair, amountA);
        TransferHelper.safeTransferFrom(tokenB, msg.sender, pair, amountB);
        liquidity = IUniswapV2Pair(pair).mint(to);
    }
    function addLiquidityETH(
        address token,
        uint amountTokenDesired,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) external virtual override payable ensure(deadline) returns (uint amou
        (amountToken, amountETH) = _addLiquidity(
            token,
            WETH,
            amountTokenDesired,
            msg.value,
            amountTokenMin,
            amountETHMin
        );
        address pair = UniswapV2Library.pairFor(factory, token, WETH);
        TransferHelper.safeTransferFrom(token, msg.sender, pair, amountToke
        IWETH(WETH).deposit{value: amountETH}();
        assert(IWETH(WETH).transfer(pair, amountETH));
        liquidity = IUniswapV2Pair(pair).mint(to);
        // refund dust eth, if any
        if (msg.value > amountETH) TransferHelper.safeTransferETH(msg.sende
    }

    // **** REMOVE LIQUIDITY ****
    function removeLiquidity(
        address tokenA,
        address tokenB,
        uint liquidity,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline
    ) public virtual override ensure(deadline) returns (uint amountA, uint
        address pair = UniswapV2Library.pairFor(factory, tokenA, tokenB);
        IUniswapV2Pair(pair).transferFrom(msg.sender, pair, liquidity); //
        (uint amount0, uint amount1) = IUniswapV2Pair(pair).burn(to);
        (address token0,) = UniswapV2Library.sortTokens(tokenA, tokenB);
        (amountA, amountB) = tokenA == token0 ? (amount0, amount1) : (amoun
        require(amountA >= amountAMin, 'UniswapV2Router: INSUFFICIENT_A_AMO
        require(amountB >= amountBMin, 'UniswapV2Router: INSUFFICIENT_B_AMO
    }
    function removeLiquidityETH(
        address token,
```

```
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) public virtual override ensure(deadline) returns (uint amountToken, u
        (amountToken, amountETH) = removeLiquidity(
            token,
            WETH,
            liquidity,
            amountTokenMin,
            amountETHMin,
            address(this),
            deadline
        );
        TransferHelper.safeTransfer(token, to, amountToken);
        IWETH(WETH).withdraw(amountETH);
        TransferHelper.safeTransferETH(to, amountETH);
    }
    function removeLiquidityWithPermit(
        address tokenA,
        address tokenB,
        uint liquidity,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline,
        bool approveMax, uint8 v, bytes32 r, bytes32 s
    ) external virtual override returns (uint amountA, uint amountB) {
        address pair = UniswapV2Library.pairFor(factory, tokenA, tokenB);
        uint value = approveMax ? uint(-1) : liquidity;
        IUniswapV2Pair(pair).permit(msg.sender, address(this), value, deadl
        (amountA, amountB) = removeLiquidity(tokenA, tokenB, liquidity, amo
    }
    function removeLiquidityETHWithPermit(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline,
        bool approveMax, uint8 v, bytes32 r, bytes32 s
    ) external virtual override returns (uint amountToken, uint amountETH)
        address pair = UniswapV2Library.pairFor(factory, token, WETH);
        uint value = approveMax ? uint(-1) : liquidity;
        IUniswapV2Pair(pair).permit(msg.sender, address(this), value, deadl
        (amountToken, amountETH) = removeLiquidityETH(token, liquidity, amo
    }

    // **** REMOVE LIQUIDITY (supporting fee-on-transfer tokens) ****
    function removeLiquidityETHSupportingFeeOnTransferTokens(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
```

```
    ) public virtual override ensure(deadline) returns (uint amountETH) {
        (, amountETH) = removeLiquidity(
            token,
            WETH,
            liquidity,
            amountTokenMin,
            amountETHMin,
            address(this),
            deadline
        );
        TransferHelper.safeTransfer(token, to, IERC20(token).balanceOf(addr
        IWETH(WETH).withdraw(amountETH);
        TransferHelper.safeTransferETH(to, amountETH);
    }
    function removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline,
        bool approveMax, uint8 v, bytes32 r, bytes32 s
    ) external virtual override returns (uint amountETH) {
        address pair = UniswapV2Library.pairFor(factory, token, WETH);
        uint value = approveMax ? uint(-1) : liquidity;
        IUniswapV2Pair(pair).permit(msg.sender, address(this), value, deadl
        amountETH = removeLiquidityETHSupportingFeeOnTransferTokens(
            token, liquidity, amountTokenMin, amountETHMin, to, deadline
        );
    }

    // **** SWAP ****
    // requires the initial amount to have already been sent to the first p
    function _swap(uint[] memory amounts, address[] memory path, address _t
        for (uint i; i < path.length - 1; i++) {
            (address input, address output) = (path[i], path[i + 1]);
            (address token0,) = UniswapV2Library.sortTokens(input, output);
            uint amountOut = amounts[i + 1];
            (uint amount0Out, uint amount1Out) = input == token0 ? (uint(0)
            address to = i < path.length - 2 ? UniswapV2Library.pairFor(fac
            IUniswapV2Pair(UniswapV2Library.pairFor(factory, input, output)
                amount0Out, amount1Out, to, new bytes(0)
            );
        }
    }
    function swapExactTokensForTokens(
        uint amountIn,
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    ) external virtual override ensure(deadline) returns (uint[] memory amo
        amounts = UniswapV2Library.getAmountsOut(factory, amountIn, path);
        require(amounts[amounts.length - 1] >= amountOutMin, 'UniswapV2Rout
        TransferHelper.safeTransferFrom(
            path[0], msg.sender, UniswapV2Library.pairFor(factory, path[0],
        );
```

```
        _swap(amounts, path, to);
    }
    function swapTokensForExactTokens(
        uint amountOut,
        uint amountInMax,
        address[] calldata path,
        address to,
        uint deadline
    ) external virtual override ensure(deadline) returns (uint[] memory amo
        amounts = UniswapV2Library.getAmountsIn(factory, amountOut, path);
        require(amounts[0] <= amountInMax, 'UniswapV2Router: EXCESSIVE_INPU
        TransferHelper.safeTransferFrom(
            path[0], msg.sender, UniswapV2Library.pairFor(factory, path[0],
        );
        _swap(amounts, path, to);
    }
    function swapExactETHForTokens(uint amountOutMin, address[] calldata pa
        external
        virtual
        override
        payable
        ensure(deadline)
        returns (uint[] memory amounts)
    {
        require(path[0] == WETH, 'UniswapV2Router: INVALID_PATH');
        amounts = UniswapV2Library.getAmountsOut(factory, msg.value, path);
        require(amounts[amounts.length - 1] >= amountOutMin, 'UniswapV2Rout
        IWETH(WETH).deposit{value: amounts[0]}();
        assert(IWETH(WETH).transfer(UniswapV2Library.pairFor(factory, path[
        _swap(amounts, path, to);
    }
    function swapTokensForExactETH(uint amountOut, uint amountInMax, addres
        external
        virtual
        override
        ensure(deadline)
        returns (uint[] memory amounts)
    {
        require(path[path.length - 1] == WETH, 'UniswapV2Router: INVALID_PA
        amounts = UniswapV2Library.getAmountsIn(factory, amountOut, path);
        require(amounts[0] <= amountInMax, 'UniswapV2Router: EXCESSIVE_INPU
        TransferHelper.safeTransferFrom(
            path[0], msg.sender, UniswapV2Library.pairFor(factory, path[0],
        );
        _swap(amounts, path, address(this));
        IWETH(WETH).withdraw(amounts[amounts.length - 1]);
        TransferHelper.safeTransferETH(to, amounts[amounts.length - 1]);
    }
    function swapExactTokensForETH(uint amountIn, uint amountOutMin, addres
        external
        virtual
        override
        ensure(deadline)
        returns (uint[] memory amounts)
    {
        require(path[path.length - 1] == WETH, 'UniswapV2Router: INVALID_PA
        amounts = UniswapV2Library.getAmountsOut(factory, amountIn, path);
```

```
        require(amounts[amounts.length - 1] >= amountOutMin, 'UniswapV2Rout
        TransferHelper.safeTransferFrom(
            path[0], msg.sender, UniswapV2Library.pairFor(factory, path[0],
        );
        _swap(amounts, path, address(this));
        IWETH(WETH).withdraw(amounts[amounts.length - 1]);
        TransferHelper.safeTransferETH(to, amounts[amounts.length - 1]);
    }
    function swapETHForExactTokens(uint amountOut, address[] calldata path,
        external
        virtual
        override
        payable
        ensure(deadline)
        returns (uint[] memory amounts)
    {
        require(path[0] == WETH, 'UniswapV2Router: INVALID_PATH');
        amounts = UniswapV2Library.getAmountsIn(factory, amountOut, path);
        require(amounts[0] <= msg.value, 'UniswapV2Router: EXCESSIVE_INPUT_
        IWETH(WETH).deposit{value: amounts[0]}();
        assert(IWETH(WETH).transfer(UniswapV2Library.pairFor(factory, path[
        _swap(amounts, path, to);
        // refund dust eth, if any
        if (msg.value > amounts[0]) TransferHelper.safeTransferETH(msg.send
    }

    // **** SWAP (supporting fee-on-transfer tokens) ****
    // requires the initial amount to have already been sent to the first p
    function _swapSupportingFeeOnTransferTokens(address[] memory path, addr
        for (uint i; i < path.length - 1; i++) {
            (address input, address output) = (path[i], path[i + 1]);
            (address token0,) = UniswapV2Library.sortTokens(input, output);
            IUniswapV2Pair pair = IUniswapV2Pair(UniswapV2Library.pairFor(f
            uint amountInput;
            uint amountOutput;
            { // scope to avoid stack too deep errors
            (uint reserve0, uint reserve1,) = pair.getReserves();
            (uint reserveInput, uint reserveOutput) = input == token0 ? (re
            amountInput = IERC20(input).balanceOf(address(pair)).sub(reserv
            amountOutput = UniswapV2Library.getAmountOut(amountInput, reser
            }
            (uint amount0Out, uint amount1Out) = input == token0 ? (uint(0)
            address to = i < path.length - 2 ? UniswapV2Library.pairFor(fac
            pair.swap(amount0Out, amount1Out, to, new bytes(0));
        }
    }
    function swapExactTokensForTokensSupportingFeeOnTransferTokens(
        uint amountIn,
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    ) external virtual override ensure(deadline) {
        TransferHelper.safeTransferFrom(
            path[0], msg.sender, UniswapV2Library.pairFor(factory, path[0],
        );
        uint balanceBefore = IERC20(path[path.length - 1]).balanceOf(to);
```

```
        _swapSupportingFeeOnTransferTokens(path, to);
        require(
            IERC20(path[path.length - 1]).balanceOf(to).sub(balanceBefore)
            'UniswapV2Router: INSUFFICIENT_OUTPUT_AMOUNT'
        );
    }
    function swapExactETHForTokensSupportingFeeOnTransferTokens(
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    )
        external
        virtual
        override
        payable
        ensure(deadline)
    {
        require(path[0] == WETH, 'UniswapV2Router: INVALID_PATH');
        uint amountIn = msg.value;
        IWETH(WETH).deposit{value: amountIn}();
        assert(IWETH(WETH).transfer(UniswapV2Library.pairFor(factory, path[
        uint balanceBefore = IERC20(path[path.length - 1]).balanceOf(to);
        _swapSupportingFeeOnTransferTokens(path, to);
        require(
            IERC20(path[path.length - 1]).balanceOf(to).sub(balanceBefore)
            'UniswapV2Router: INSUFFICIENT_OUTPUT_AMOUNT'
        );
    }
    function swapExactTokensForETHSupportingFeeOnTransferTokens(
        uint amountIn,
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    )
        external
        virtual
        override
        ensure(deadline)
    {
        require(path[path.length - 1] == WETH, 'UniswapV2Router: INVALID_PA
        TransferHelper.safeTransferFrom(
            path[0], msg.sender, UniswapV2Library.pairFor(factory, path[0],
        );
        _swapSupportingFeeOnTransferTokens(path, address(this));
        uint amountOut = IERC20(WETH).balanceOf(address(this));
        require(amountOut >= amountOutMin, 'UniswapV2Router: INSUFFICIENT_O
        IWETH(WETH).withdraw(amountOut);
        TransferHelper.safeTransferETH(to, amountOut);
    }

    // **** LIBRARY FUNCTIONS ****
    function quote(uint amountA, uint reserveA, uint reserveB) public pure
        return UniswapV2Library.quote(amountA, reserveA, reserveB);
    }
```

```
    function getAmountOut(uint amountIn, uint reserveIn, uint reserveOut)
        public
        pure
        virtual
        override
        returns (uint amountOut)
    {
        return UniswapV2Library.getAmountOut(amountIn, reserveIn, reserveOu
    }

    function getAmountIn(uint amountOut, uint reserveIn, uint reserveOut)
        public
        pure
        virtual
        override
        returns (uint amountIn)
    {
        return UniswapV2Library.getAmountIn(amountOut, reserveIn, reserveOu
    }

    function getAmountsOut(uint amountIn, address[] memory path)
        public
        view
        virtual
        override
        returns (uint[] memory amounts)
    {
        return UniswapV2Library.getAmountsOut(factory, amountIn, path);
    }

    function getAmountsIn(uint amountOut, address[] memory path)
        public
        view
        virtual
        override
        returns (uint[] memory amounts)
    {
        return UniswapV2Library.getAmountsIn(factory, amountOut, path);
    }
}

// a library for performing overflow-safe math, courtesy of DappHub (https:

library SafeMath {
    function add(uint x, uint y) internal pure returns (uint z) {
        require((z = x + y) >= x, 'ds-math-add-overflow');
    }

    function sub(uint x, uint y) internal pure returns (uint z) {
        require((z = x - y) <= x, 'ds-math-sub-underflow');
    }

    function mul(uint x, uint y) internal pure returns (uint z) {
        require(y == 0 || (z = x * y) / y == x, 'ds-math-mul-overflow');
    }
}
```

```
library UniswapV2Library {
    using SafeMath for uint;

    // returns sorted token addresses, used to handle return values from pa
    function sortTokens(address tokenA, address tokenB) internal pure retur
        require(tokenA != tokenB, 'UniswapV2Library: IDENTICAL_ADDRESSES');
        (token0, token1) = tokenA < tokenB ? (tokenA, tokenB) : (tokenB, to
        require(token0 != address(0), 'UniswapV2Library: ZERO_ADDRESS');
    }

    // calculates the CREATE2 address for a pair without making any externa
    function pairFor(address factory, address tokenA, address tokenB) inter
        (address token0, address token1) = sortTokens(tokenA, tokenB);
        pair = address(uint(keccak256(abi.encodePacked(
                hex'ff',
                factory,
                keccak256(abi.encodePacked(token0, token1)),
                hex'96e8ac4277198ff8b6f785478aa9a39f403cb768dd02cbee326c3e7
            ))));
    }

    // fetches and sorts the reserves for a pair
    function getReserves(address factory, address tokenA, address tokenB) i
        (address token0,) = sortTokens(tokenA, tokenB);
        (uint reserve0, uint reserve1,) = IUniswapV2Pair(pairFor(factory, t
        (reserveA, reserveB) = tokenA == token0 ? (reserve0, reserve1) : (r
    }

    // given some amount of an asset and pair reserves, returns an equivale
    function quote(uint amountA, uint reserveA, uint reserveB) internal pur
        require(amountA > 0, 'UniswapV2Library: INSUFFICIENT_AMOUNT');
        require(reserveA > 0 && reserveB > 0, 'UniswapV2Library: INSUFFICIE
        amountB = amountA.mul(reserveB) / reserveA;
    }

    // given an input amount of an asset and pair reserves, returns the max
    function getAmountOut(uint amountIn, uint reserveIn, uint reserveOut) i
        require(amountIn > 0, 'UniswapV2Library: INSUFFICIENT_INPUT_AMOUNT'
        require(reserveIn > 0 && reserveOut > 0, 'UniswapV2Library: INSUFFI
        uint amountInWithFee = amountIn.mul(997);
        uint numerator = amountInWithFee.mul(reserveOut);
        uint denominator = reserveIn.mul(1000).add(amountInWithFee);
        amountOut = numerator / denominator;
    }

    // given an output amount of an asset and pair reserves, returns a requ
    function getAmountIn(uint amountOut, uint reserveIn, uint reserveOut) i
        require(amountOut > 0, 'UniswapV2Library: INSUFFICIENT_OUTPUT_AMOUN
        require(reserveIn > 0 && reserveOut > 0, 'UniswapV2Library: INSUFFI
        uint numerator = reserveIn.mul(amountOut).mul(1000);
        uint denominator = reserveOut.sub(amountOut).mul(997);
        amountIn = (numerator / denominator).add(1);
    }

    // performs chained getAmountOut calculations on any number of pairs
    function getAmountsOut(address factory, uint amountIn, address[] memory
        require(path.length >= 2, 'UniswapV2Library: INVALID_PATH');
```

```
        amounts = new uint[](path.length);
        amounts[0] = amountIn;
        for (uint i; i < path.length - 1; i++) {
            (uint reserveIn, uint reserveOut) = getReserves(factory, path[i
            amounts[i + 1] = getAmountOut(amounts[i], reserveIn, reserveOut
        }
    }

    // performs chained getAmountIn calculations on any number of pairs
    function getAmountsIn(address factory, uint amountOut, address[] memory
        require(path.length >= 2, 'UniswapV2Library: INVALID_PATH');
        amounts = new uint[](path.length);
        amounts[amounts.length - 1] = amountOut;
        for (uint i = path.length - 1; i > 0; i--) {
            (uint reserveIn, uint reserveOut) = getReserves(factory, path[i
            amounts[i - 1] = getAmountIn(amounts[i], reserveIn, reserveOut)
        }
    }
}

// helper methods for interacting with ERC20 tokens and sending ETH that do
library TransferHelper {
    function safeApprove(address token, address to, uint value) internal {
        // bytes4(keccak256(bytes('approve(address,uint256)')));
        (bool success, bytes memory data) = token.call(abi.encodeWithSelect
        require(success && (data.length == 0 || abi.decode(data, (bool))),
    }

    function safeTransfer(address token, address to, uint value) internal {
        // bytes4(keccak256(bytes('transfer(address,uint256)')));
        (bool success, bytes memory data) = token.call(abi.encodeWithSelect
        require(success && (data.length == 0 || abi.decode(data, (bool))),
    }

    function safeTransferFrom(address token, address from, address to, uint
        // bytes4(keccak256(bytes('transferFrom(address,address,uint256)'))
        (bool success, bytes memory data) = token.call(abi.encodeWithSelect
        require(success && (data.length == 0 || abi.decode(data, (bool))),
    }

    function safeTransferETH(address to, uint value) internal {
        (bool success,) = to.call{value:value}(new bytes(0));
        require(success, 'TransferHelper: ETH_TRANSFER_FAILED');
    }
}"""


pancakeV2RouterCode = """
/**
 *Submitted for verification at BscScan.com on 2021-04-23
*/

// File: @uniswap\lib\contracts\libraries\TransferHelper.sol

pragma solidity >=0.6.0;

// helper methods for interacting with ERC20 tokens and sending ETH that do
```

```
library TransferHelper {
    function safeApprove(address token, address to, uint value) internal {
        // bytes4(keccak256(bytes('approve(address,uint256)')));
        (bool success, bytes memory data) = token.call(abi.encodeWithSelect
        require(success && (data.length == 0 || abi.decode(data, (bool))),
    }

    function safeTransfer(address token, address to, uint value) internal {
        // bytes4(keccak256(bytes('transfer(address,uint256)')));
        (bool success, bytes memory data) = token.call(abi.encodeWithSelect
        require(success && (data.length == 0 || abi.decode(data, (bool))),
    }

    function safeTransferFrom(address token, address from, address to, uint
        // bytes4(keccak256(bytes('transferFrom(address,address,uint256)'))
        (bool success, bytes memory data) = token.call(abi.encodeWithSelect
        require(success && (data.length == 0 || abi.decode(data, (bool))),
    }

    function safeTransferETH(address to, uint value) internal {
        (bool success,) = to.call{value:value}(new bytes(0));
        require(success, 'TransferHelper: ETH_TRANSFER_FAILED');
    }
}

// File: contracts\interfaces\IPancakeRouter01.sol

pragma solidity >=0.6.2;

interface IPancakeRouter01 {
    function factory() external pure returns (address);
    function WETH() external pure returns (address);

    function addLiquidity(
        address tokenA,
        address tokenB,
        uint amountADesired,
        uint amountBDesired,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline
    ) external returns (uint amountA, uint amountB, uint liquidity);
    function addLiquidityETH(
        address token,
        uint amountTokenDesired,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) external payable returns (uint amountToken, uint amountETH, uint liqu
    function removeLiquidity(
        address tokenA,
        address tokenB,
        uint liquidity,
        uint amountAMin,
        uint amountBMin,
```

```
        address to,
        uint deadline
    ) external returns (uint amountA, uint amountB);
    function removeLiquidityETH(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) external returns (uint amountToken, uint amountETH);
    function removeLiquidityWithPermit(
        address tokenA,
        address tokenB,
        uint liquidity,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline,
        bool approveMax, uint8 v, bytes32 r, bytes32 s
    ) external returns (uint amountA, uint amountB);
    function removeLiquidityETHWithPermit(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline,
        bool approveMax, uint8 v, bytes32 r, bytes32 s
    ) external returns (uint amountToken, uint amountETH);
    function swapExactTokensForTokens(
        uint amountIn,
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    ) external returns (uint[] memory amounts);
    function swapTokensForExactTokens(
        uint amountOut,
        uint amountInMax,
        address[] calldata path,
        address to,
        uint deadline
    ) external returns (uint[] memory amounts);
    function swapExactETHForTokens(uint amountOutMin, address[] calldata pa
        external
        payable
        returns (uint[] memory amounts);
    function swapTokensForExactETH(uint amountOut, uint amountInMax, addres
        external
        returns (uint[] memory amounts);
    function swapExactTokensForETH(uint amountIn, uint amountOutMin, addres
        external
        returns (uint[] memory amounts);
    function swapETHForExactTokens(uint amountOut, address[] calldata path,
        external
        payable
```

```
        returns (uint[] memory amounts);

    function quote(uint amountA, uint reserveA, uint reserveB) external pur
    function getAmountOut(uint amountIn, uint reserveIn, uint reserveOut) e
    function getAmountIn(uint amountOut, uint reserveIn, uint reserveOut) e
    function getAmountsOut(uint amountIn, address[] calldata path) external
    function getAmountsIn(uint amountOut, address[] calldata path) external
}

// File: contracts\interfaces\IPancakeRouter02.sol

pragma solidity >=0.6.2;

interface IPancakeRouter02 is IPancakeRouter01 {
    function removeLiquidityETHSupportingFeeOnTransferTokens(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) external returns (uint amountETH);
    function removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline,
        bool approveMax, uint8 v, bytes32 r, bytes32 s
    ) external returns (uint amountETH);

    function swapExactTokensForTokensSupportingFeeOnTransferTokens(
        uint amountIn,
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    ) external;
    function swapExactETHForTokensSupportingFeeOnTransferTokens(
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    ) external payable;
    function swapExactTokensForETHSupportingFeeOnTransferTokens(
        uint amountIn,
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    ) external;
}

// File: contracts\interfaces\IPancakeFactory.sol

pragma solidity >=0.5.0;
```

```
interface IPancakeFactory {
    event PairCreated(address indexed token0, address indexed token1, addre

    function feeTo() external view returns (address);
    function feeToSetter() external view returns (address);

    function getPair(address tokenA, address tokenB) external view returns
    function allPairs(uint) external view returns (address pair);
    function allPairsLength() external view returns (uint);

    function createPair(address tokenA, address tokenB) external returns (a

    function setFeeTo(address) external;
    function setFeeToSetter(address) external;

    function INIT_CODE_PAIR_HASH() external view returns (bytes32);
}

// File: contracts\libraries\SafeMath.sol

pragma solidity =0.6.6;

// a library for performing overflow-safe math, courtesy of DappHub (https:

library SafeMath {
    function add(uint x, uint y) internal pure returns (uint z) {
        require((z = x + y) >= x, 'ds-math-add-overflow');
    }

    function sub(uint x, uint y) internal pure returns (uint z) {
        require((z = x - y) <= x, 'ds-math-sub-underflow');
    }

    function mul(uint x, uint y) internal pure returns (uint z) {
        require(y == 0 || (z = x * y) / y == x, 'ds-math-mul-overflow');
    }
}

// File: contracts\interfaces\IPancakePair.sol

pragma solidity >=0.5.0;

interface IPancakePair {
    event Approval(address indexed owner, address indexed spender, uint val
    event Transfer(address indexed from, address indexed to, uint value);

    function name() external pure returns (string memory);
    function symbol() external pure returns (string memory);
    function decimals() external pure returns (uint8);
    function totalSupply() external view returns (uint);
    function balanceOf(address owner) external view returns (uint);
    function allowance(address owner, address spender) external view return

    function approve(address spender, uint value) external returns (bool);
    function transfer(address to, uint value) external returns (bool);
    function transferFrom(address from, address to, uint value) external re
```

```
    function DOMAIN_SEPARATOR() external view returns (bytes32);
    function PERMIT_TYPEHASH() external pure returns (bytes32);
    function nonces(address owner) external view returns (uint);

    function permit(address owner, address spender, uint value, uint deadli

    event Mint(address indexed sender, uint amount0, uint amount1);
    event Burn(address indexed sender, uint amount0, uint amount1, address
    event Swap(
        address indexed sender,
        uint amount0In,
        uint amount1In,
        uint amount0Out,
        uint amount1Out,
        address indexed to
    );
    event Sync(uint112 reserve0, uint112 reserve1);

    function MINIMUM_LIQUIDITY() external pure returns (uint);
    function factory() external view returns (address);
    function token0() external view returns (address);
    function token1() external view returns (address);
    function getReserves() external view returns (uint112 reserve0, uint112
    function price0CumulativeLast() external view returns (uint);
    function price1CumulativeLast() external view returns (uint);
    function kLast() external view returns (uint);

    function mint(address to) external returns (uint liquidity);
    function burn(address to) external returns (uint amount0, uint amount1)
    function swap(uint amount0Out, uint amount1Out, address to, bytes calld
    function skim(address to) external;
    function sync() external;

    function initialize(address, address) external;
}

// File: contracts\libraries\PancakeLibrary.sol

pragma solidity >=0.5.0;



library PancakeLibrary {
    using SafeMath for uint;

    // returns sorted token addresses, used to handle return values from pa
    function sortTokens(address tokenA, address tokenB) internal pure retur
        require(tokenA != tokenB, 'PancakeLibrary: IDENTICAL_ADDRESSES');
        (token0, token1) = tokenA < tokenB ? (tokenA, tokenB) : (tokenB, to
        require(token0 != address(0), 'PancakeLibrary: ZERO_ADDRESS');
    }

    // calculates the CREATE2 address for a pair without making any externa
    function pairFor(address factory, address tokenA, address tokenB) inter
        (address token0, address token1) = sortTokens(tokenA, tokenB);
        pair = address(uint(keccak256(abi.encodePacked(
```

```
            hex'ff',
            factory,
            keccak256(abi.encodePacked(token0, token1)),
            hex'00fb7f630766e6a796048ea87d01acd3068e8ff67d078148a3fa3f4
        ))));
}

// fetches and sorts the reserves for a pair
function getReserves(address factory, address tokenA, address tokenB) i
    (address token0,) = sortTokens(tokenA, tokenB);
    pairFor(factory, tokenA, tokenB);
    (uint reserve0, uint reserve1,) = IPancakePair(pairFor(factory, tok
    (reserveA, reserveB) = tokenA == token0 ? (reserve0, reserve1) : (r
}

// given some amount of an asset and pair reserves, returns an equivale
function quote(uint amountA, uint reserveA, uint reserveB) internal pur
    require(amountA > 0, 'PancakeLibrary: INSUFFICIENT_AMOUNT');
    require(reserveA > 0 && reserveB > 0, 'PancakeLibrary: INSUFFICIENT
    amountB = amountA.mul(reserveB) / reserveA;
}

// given an input amount of an asset and pair reserves, returns the max
function getAmountOut(uint amountIn, uint reserveIn, uint reserveOut) i
    require(amountIn > 0, 'PancakeLibrary: INSUFFICIENT_INPUT_AMOUNT');
    require(reserveIn > 0 && reserveOut > 0, 'PancakeLibrary: INSUFFICI
    uint amountInWithFee = amountIn.mul(9975);
    uint numerator = amountInWithFee.mul(reserveOut);
    uint denominator = reserveIn.mul(10000).add(amountInWithFee);
    amountOut = numerator / denominator;
}

// given an output amount of an asset and pair reserves, returns a requ
function getAmountIn(uint amountOut, uint reserveIn, uint reserveOut) i
    require(amountOut > 0, 'PancakeLibrary: INSUFFICIENT_OUTPUT_AMOUNT'
    require(reserveIn > 0 && reserveOut > 0, 'PancakeLibrary: INSUFFICI
    uint numerator = reserveIn.mul(amountOut).mul(10000);
    uint denominator = reserveOut.sub(amountOut).mul(9975);
    amountIn = (numerator / denominator).add(1);
}

// performs chained getAmountOut calculations on any number of pairs
function getAmountsOut(address factory, uint amountIn, address[] memory
    require(path.length >= 2, 'PancakeLibrary: INVALID_PATH');
    amounts = new uint[](path.length);
    amounts[0] = amountIn;
    for (uint i; i < path.length - 1; i++) {
        (uint reserveIn, uint reserveOut) = getReserves(factory, path[i
        amounts[i + 1] = getAmountOut(amounts[i], reserveIn, reserveOut
    }
}

// performs chained getAmountIn calculations on any number of pairs
function getAmountsIn(address factory, uint amountOut, address[] memory
    require(path.length >= 2, 'PancakeLibrary: INVALID_PATH');
    amounts = new uint[](path.length);
    amounts[amounts.length - 1] = amountOut;
```

```
        for (uint i = path.length - 1; i > 0; i--) {
            (uint reserveIn, uint reserveOut) = getReserves(factory, path[i
            amounts[i - 1] = getAmountIn(amounts[i], reserveIn, reserveOut)
        }
    }
}

// File: contracts\interfaces\IERC20.sol

pragma solidity >=0.5.0;

interface IERC20 {
    event Approval(address indexed owner, address indexed spender, uint val
    event Transfer(address indexed from, address indexed to, uint value);

    function name() external view returns (string memory);
    function symbol() external view returns (string memory);
    function decimals() external view returns (uint8);
    function totalSupply() external view returns (uint);
    function balanceOf(address owner) external view returns (uint);
    function allowance(address owner, address spender) external view return

    function approve(address spender, uint value) external returns (bool);
    function transfer(address to, uint value) external returns (bool);
    function transferFrom(address from, address to, uint value) external re
}

// File: contracts\interfaces\IWETH.sol

pragma solidity >=0.5.0;

interface IWETH {
    function deposit() external payable;
    function transfer(address to, uint value) external returns (bool);
    function withdraw(uint) external;
}

// File: contracts\PancakeRouter.sol

pragma solidity =0.6.6;









contract PancakeRouter is IPancakeRouter02 {
    using SafeMath for uint;

    address public immutable override factory;
    address public immutable override WETH;

    modifier ensure(uint deadline) {
        require(deadline >= block.timestamp, 'PancakeRouter: EXPIRED');
        _;
    }
```

```
    constructor(address _factory, address _WETH) public {
        factory = _factory;
        WETH = _WETH;
    }

    receive() external payable {
        assert(msg.sender == WETH); // only accept ETH via fallback from th
    }

    // **** ADD LIQUIDITY ****
    function _addLiquidity(
        address tokenA,
        address tokenB,
        uint amountADesired,
        uint amountBDesired,
        uint amountAMin,
        uint amountBMin
    ) internal virtual returns (uint amountA, uint amountB) {
        // create the pair if it doesn't exist yet
        if (IPancakeFactory(factory).getPair(tokenA, tokenB) == address(0))
            IPancakeFactory(factory).createPair(tokenA, tokenB);
        }
        (uint reserveA, uint reserveB) = PancakeLibrary.getReserves(factory
        if (reserveA == 0 && reserveB == 0) {
            (amountA, amountB) = (amountADesired, amountBDesired);
        } else {
            uint amountBOptimal = PancakeLibrary.quote(amountADesired, rese
            if (amountBOptimal <= amountBDesired) {
                require(amountBOptimal >= amountBMin, 'PancakeRouter: INSUF
                (amountA, amountB) = (amountADesired, amountBOptimal);
            } else {
                uint amountAOptimal = PancakeLibrary.quote(amountBDesired,
                assert(amountAOptimal <= amountADesired);
                require(amountAOptimal >= amountAMin, 'PancakeRouter: INSUF
                (amountA, amountB) = (amountAOptimal, amountBDesired);
            }
        }
    }
    function addLiquidity(
        address tokenA,
        address tokenB,
        uint amountADesired,
        uint amountBDesired,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline
    ) external virtual override ensure(deadline) returns (uint amountA, uin
        (amountA, amountB) = _addLiquidity(tokenA, tokenB, amountADesired,
        address pair = PancakeLibrary.pairFor(factory, tokenA, tokenB);
        TransferHelper.safeTransferFrom(tokenA, msg.sender, pair, amountA);
        TransferHelper.safeTransferFrom(tokenB, msg.sender, pair, amountB);
        liquidity = IPancakePair(pair).mint(to);
    }
    function addLiquidityETH(
        address token,
```

```
        uint amountTokenDesired,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) external virtual override payable ensure(deadline) returns (uint amou
        (amountToken, amountETH) = _addLiquidity(
            token,
            WETH,
            amountTokenDesired,
            msg.value,
            amountTokenMin,
            amountETHMin
        );
        address pair = PancakeLibrary.pairFor(factory, token, WETH);
        TransferHelper.safeTransferFrom(token, msg.sender, pair, amountToke
        IWETH(WETH).deposit{value: amountETH}();
        assert(IWETH(WETH).transfer(pair, amountETH));
        liquidity = IPancakePair(pair).mint(to);
        // refund dust eth, if any
        if (msg.value > amountETH) TransferHelper.safeTransferETH(msg.sende
    }

    // **** REMOVE LIQUIDITY ****
    function removeLiquidity(
        address tokenA,
        address tokenB,
        uint liquidity,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline
    ) public virtual override ensure(deadline) returns (uint amountA, uint
        address pair = PancakeLibrary.pairFor(factory, tokenA, tokenB);
        IPancakePair(pair).transferFrom(msg.sender, pair, liquidity); // se
        (uint amount0, uint amount1) = IPancakePair(pair).burn(to);
        (address token0,) = PancakeLibrary.sortTokens(tokenA, tokenB);
        (amountA, amountB) = tokenA == token0 ? (amount0, amount1) : (amoun
        require(amountA >= amountAMin, 'PancakeRouter: INSUFFICIENT_A_AMOUN
        require(amountB >= amountBMin, 'PancakeRouter: INSUFFICIENT_B_AMOUN
    }
    function removeLiquidityETH(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) public virtual override ensure(deadline) returns (uint amountToken, u
        (amountToken, amountETH) = removeLiquidity(
            token,
            WETH,
            liquidity,
            amountTokenMin,
            amountETHMin,
            address(this),
            deadline
```

```
        );
        TransferHelper.safeTransfer(token, to, amountToken);
        IWETH(WETH).withdraw(amountETH);
        TransferHelper.safeTransferETH(to, amountETH);
    }
    function removeLiquidityWithPermit(
        address tokenA,
        address tokenB,
        uint liquidity,
        uint amountAMin,
        uint amountBMin,
        address to,
        uint deadline,
        bool approveMax, uint8 v, bytes32 r, bytes32 s
    ) external virtual override returns (uint amountA, uint amountB) {
        address pair = PancakeLibrary.pairFor(factory, tokenA, tokenB);
        uint value = approveMax ? uint(-1) : liquidity;
        IPancakePair(pair).permit(msg.sender, address(this), value, deadlin
        (amountA, amountB) = removeLiquidity(tokenA, tokenB, liquidity, amo
    }
    function removeLiquidityETHWithPermit(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline,
        bool approveMax, uint8 v, bytes32 r, bytes32 s
    ) external virtual override returns (uint amountToken, uint amountETH)
        address pair = PancakeLibrary.pairFor(factory, token, WETH);
        uint value = approveMax ? uint(-1) : liquidity;
        IPancakePair(pair).permit(msg.sender, address(this), value, deadlin
        (amountToken, amountETH) = removeLiquidityETH(token, liquidity, amo
    }

    // **** REMOVE LIQUIDITY (supporting fee-on-transfer tokens) ****
    function removeLiquidityETHSupportingFeeOnTransferTokens(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline
    ) public virtual override ensure(deadline) returns (uint amountETH) {
        (, amountETH) = removeLiquidity(
            token,
            WETH,
            liquidity,
            amountTokenMin,
            amountETHMin,
            address(this),
            deadline
        );
        TransferHelper.safeTransfer(token, to, IERC20(token).balanceOf(addr
        IWETH(WETH).withdraw(amountETH);
        TransferHelper.safeTransferETH(to, amountETH);
    }
```

```
    function removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(
        address token,
        uint liquidity,
        uint amountTokenMin,
        uint amountETHMin,
        address to,
        uint deadline,
        bool approveMax, uint8 v, bytes32 r, bytes32 s
    ) external virtual override returns (uint amountETH) {
        address pair = PancakeLibrary.pairFor(factory, token, WETH);
        uint value = approveMax ? uint(-1) : liquidity;
        IPancakePair(pair).permit(msg.sender, address(this), value, deadlin
        amountETH = removeLiquidityETHSupportingFeeOnTransferTokens(
            token, liquidity, amountTokenMin, amountETHMin, to, deadline
        );
    }

    // **** SWAP ****
    // requires the initial amount to have already been sent to the first p
    function _swap(uint[] memory amounts, address[] memory path, address _t
        for (uint i; i < path.length - 1; i++) {
            (address input, address output) = (path[i], path[i + 1]);
            (address token0,) = PancakeLibrary.sortTokens(input, output);
            uint amountOut = amounts[i + 1];
            (uint amount0Out, uint amount1Out) = input == token0 ? (uint(0)
            address to = i < path.length - 2 ? PancakeLibrary.pairFor(facto
            IPancakePair(PancakeLibrary.pairFor(factory, input, output)).sw
                amount0Out, amount1Out, to, new bytes(0)
            );
        }
    }
    function swapExactTokensForTokens(
        uint amountIn,
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    ) external virtual override ensure(deadline) returns (uint[] memory amo
        amounts = PancakeLibrary.getAmountsOut(factory, amountIn, path);
        require(amounts[amounts.length - 1] >= amountOutMin, 'PancakeRouter
        TransferHelper.safeTransferFrom(
            path[0], msg.sender, PancakeLibrary.pairFor(factory, path[0], p
        );
        _swap(amounts, path, to);
    }
    function swapTokensForExactTokens(
        uint amountOut,
        uint amountInMax,
        address[] calldata path,
        address to,
        uint deadline
    ) external virtual override ensure(deadline) returns (uint[] memory amo
        amounts = PancakeLibrary.getAmountsIn(factory, amountOut, path);
        require(amounts[0] <= amountInMax, 'PancakeRouter: EXCESSIVE_INPUT_
        TransferHelper.safeTransferFrom(
            path[0], msg.sender, PancakeLibrary.pairFor(factory, path[0], p
        );
```

```
        _swap(amounts, path, to);
    }
    function swapExactETHForTokens(uint amountOutMin, address[] calldata pa
        external
        virtual
        override
        payable
        ensure(deadline)
        returns (uint[] memory amounts)
    {
        require(path[0] == WETH, 'PancakeRouter: INVALID_PATH');
        amounts = PancakeLibrary.getAmountsOut(factory, msg.value, path);
        require(amounts[amounts.length - 1] >= amountOutMin, 'PancakeRouter
        IWETH(WETH).deposit{value: amounts[0]}();
        assert(IWETH(WETH).transfer(PancakeLibrary.pairFor(factory, path[0]
        _swap(amounts, path, to);
    }
    function swapTokensForExactETH(uint amountOut, uint amountInMax, addres
        external
        virtual
        override
        ensure(deadline)
        returns (uint[] memory amounts)
    {
        require(path[path.length - 1] == WETH, 'PancakeRouter: INVALID_PATH
        amounts = PancakeLibrary.getAmountsIn(factory, amountOut, path);
        require(amounts[0] <= amountInMax, 'PancakeRouter: EXCESSIVE_INPUT_
        TransferHelper.safeTransferFrom(
            path[0], msg.sender, PancakeLibrary.pairFor(factory, path[0], p
        );
        _swap(amounts, path, address(this));
        IWETH(WETH).withdraw(amounts[amounts.length - 1]);
        TransferHelper.safeTransferETH(to, amounts[amounts.length - 1]);
    }
    function swapExactTokensForETH(uint amountIn, uint amountOutMin, addres
        external
        virtual
        override
        ensure(deadline)
        returns (uint[] memory amounts)
    {
        require(path[path.length - 1] == WETH, 'PancakeRouter: INVALID_PATH
        amounts = PancakeLibrary.getAmountsOut(factory, amountIn, path);
        require(amounts[amounts.length - 1] >= amountOutMin, 'PancakeRouter
        TransferHelper.safeTransferFrom(
            path[0], msg.sender, PancakeLibrary.pairFor(factory, path[0], p
        );
        _swap(amounts, path, address(this));
        IWETH(WETH).withdraw(amounts[amounts.length - 1]);
        TransferHelper.safeTransferETH(to, amounts[amounts.length - 1]);
    }
    function swapETHForExactTokens(uint amountOut, address[] calldata path,
        external
        virtual
        override
        payable
        ensure(deadline)
```

```
    returns (uint[] memory amounts)
{
    require(path[0] == WETH, 'PancakeRouter: INVALID_PATH');
    amounts = PancakeLibrary.getAmountsIn(factory, amountOut, path);
    require(amounts[0] <= msg.value, 'PancakeRouter: EXCESSIVE_INPUT_AM
    IWETH(WETH).deposit{value: amounts[0]}();
    assert(IWETH(WETH).transfer(PancakeLibrary.pairFor(factory, path[0]
    _swap(amounts, path, to);
    // refund dust eth, if any
    if (msg.value > amounts[0]) TransferHelper.safeTransferETH(msg.send
}

// **** SWAP (supporting fee-on-transfer tokens) ****
// requires the initial amount to have already been sent to the first p
function _swapSupportingFeeOnTransferTokens(address[] memory path, addr
    for (uint i; i < path.length - 1; i++) {
        (address input, address output) = (path[i], path[i + 1]);
        (address token0,) = PancakeLibrary.sortTokens(input, output);
        IPancakePair pair = IPancakePair(PancakeLibrary.pairFor(factory
        uint amountInput;
        uint amountOutput;
        { // scope to avoid stack too deep errors
        (uint reserve0, uint reserve1,) = pair.getReserves();
        (uint reserveInput, uint reserveOutput) = input == token0 ? (re
        amountInput = IERC20(input).balanceOf(address(pair)).sub(reserv
        amountOutput = PancakeLibrary.getAmountOut(amountInput, reserve
        }
        (uint amount0Out, uint amount1Out) = input == token0 ? (uint(0)
        address to = i < path.length - 2 ? PancakeLibrary.pairFor(facto
        pair.swap(amount0Out, amount1Out, to, new bytes(0));
    }
}
function swapExactTokensForTokensSupportingFeeOnTransferTokens(
    uint amountIn,
    uint amountOutMin,
    address[] calldata path,
    address to,
    uint deadline
) external virtual override ensure(deadline) {
    TransferHelper.safeTransferFrom(
        path[0], msg.sender, PancakeLibrary.pairFor(factory, path[0], p
    );
    uint balanceBefore = IERC20(path[path.length - 1]).balanceOf(to);
    _swapSupportingFeeOnTransferTokens(path, to);
    require(
        IERC20(path[path.length - 1]).balanceOf(to).sub(balanceBefore)
        'PancakeRouter: INSUFFICIENT_OUTPUT_AMOUNT'
    );
}
function swapExactETHForTokensSupportingFeeOnTransferTokens(
    uint amountOutMin,
    address[] calldata path,
    address to,
    uint deadline
)
    external
    virtual
```

```
        override
        payable
        ensure(deadline)
    {
        require(path[0] == WETH, 'PancakeRouter: INVALID_PATH');
        uint amountIn = msg.value;
        IWETH(WETH).deposit{value: amountIn}();
        assert(IWETH(WETH).transfer(PancakeLibrary.pairFor(factory, path[0]
        uint balanceBefore = IERC20(path[path.length - 1]).balanceOf(to);
        _swapSupportingFeeOnTransferTokens(path, to);
        require(
            IERC20(path[path.length - 1]).balanceOf(to).sub(balanceBefore)
            'PancakeRouter: INSUFFICIENT_OUTPUT_AMOUNT'
        );
    }
    function swapExactTokensForETHSupportingFeeOnTransferTokens(
        uint amountIn,
        uint amountOutMin,
        address[] calldata path,
        address to,
        uint deadline
    )
        external
        virtual
        override
        ensure(deadline)
    {
        require(path[path.length - 1] == WETH, 'PancakeRouter: INVALID_PATH
        TransferHelper.safeTransferFrom(
            path[0], msg.sender, PancakeLibrary.pairFor(factory, path[0], p
        );
        _swapSupportingFeeOnTransferTokens(path, address(this));
        uint amountOut = IERC20(WETH).balanceOf(address(this));
        require(amountOut >= amountOutMin, 'PancakeRouter: INSUFFICIENT_OUT
        IWETH(WETH).withdraw(amountOut);
        TransferHelper.safeTransferETH(to, amountOut);
    }

    // **** LIBRARY FUNCTIONS ****
    function quote(uint amountA, uint reserveA, uint reserveB) public pure
        return PancakeLibrary.quote(amountA, reserveA, reserveB);
    }

    function getAmountOut(uint amountIn, uint reserveIn, uint reserveOut)
        public
        pure
        virtual
        override
        returns (uint amountOut)
    {
        return PancakeLibrary.getAmountOut(amountIn, reserveIn, reserveOut)
    }

    function getAmountIn(uint amountOut, uint reserveIn, uint reserveOut)
        public
        pure
        virtual
```

```
        override
        returns (uint amountIn)
    {
        return PancakeLibrary.getAmountIn(amountOut, reserveIn, reserveOut);
    }

    function getAmountsOut(uint amountIn, address[] memory path)
        public
        view
        virtual
        override
        returns (uint[] memory amounts)
    {
        return PancakeLibrary.getAmountsOut(factory, amountIn, path);
    }

    function getAmountsIn(uint amountOut, address[] memory path)
        public
        view
        virtual
        override
        returns (uint[] memory amounts)
    {
        return PancakeLibrary.getAmountsIn(factory, amountOut, path);
    }
}
"""
```

Compare both the codes as string using jaro distance metric

In [4]:
```
code_similarity_percentage = jellyfish.jaro_distance(uniswapV2RouterCode,pa
```

In [5]:
```
print(code_similarity_percentage)
```

```
82.84083652609377
```

Compare if both the contracts contain the same functions

In [6]:
```
uniswap_router = w3.eth.contract(address=uniswapV2RouterAddress, abi=uniswa
uniswap_functions_list = list(uniswap_router.functions)
```

In [7]:
```
panacke_router = w3.eth.contract(address=pancakeV2RouterAddress, abi=pancak
pancake_functions_list = list(panacke_router.functions)
```

In [8]:
```
functions_similarity_percentage = len(set(uniswap_functions_list) & set(pan
```

In [9]:
```
print(functions_similarity_percentage)
```

```
100.0
```

In [ ]:

In [ ]:

In [ ]: