Galaga Token $GLG

Introduction

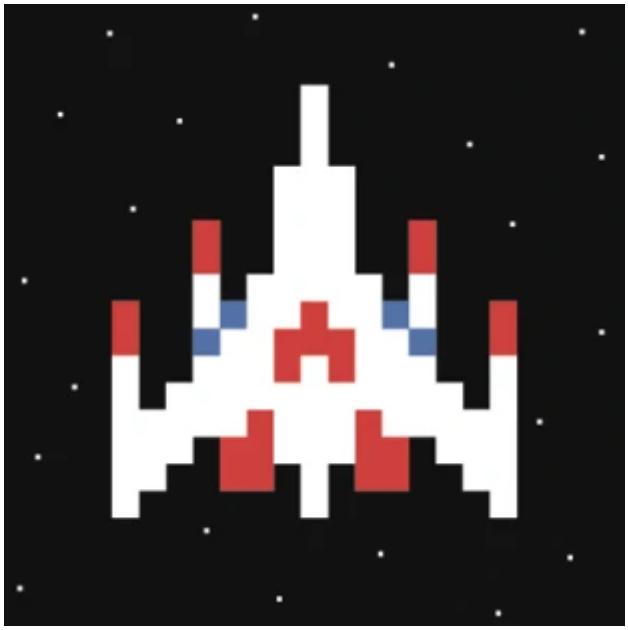
About Us

WE LOVE GALAGA SO MUCH WE MEMED IT on BSC

Galaga was a viral hit when it burst into arcades back in good ol' 1981. The sequel to 1979's Galaxian, Galaga, which also spawned several sequels of its own, is an intense space shooter. Ported to a bounty of consoles, from the Atari 7800 to Xbox 360, it is likely due to some act of divine intervention if you have yet to play this game. From the so-called Golden Age of Arcade Games, Galaga still enjoys widespread popularity today amongst vintage, rebuilt, and replica coin-operated machines.

Buy Galagatoken $GLG BSC on Pancakeswap

Contract: 0x5448d9fc695b5f1e244c374cf4b398770d788f96



Fees

- 2% of each transaction is automatically added to the liquidity pool.
- 1% of each transaction will be automatically given to the Token Holders.
- 2% token sent to the Marketing and Developement wallet.
- Receive address to Burn 1% Transaction Fees daily after Initial Launch

Galagatoken $GLG - Contract ID -0x5448d9fc695b5f1e244c374cf4b398770d788f96

-

Tokenomics

Initial Token Supply 100,000,000,000,000 $GLG (Galagatoken)

25% Burn TX Hash on bscscan

10% For Airdrop with details to be announced soon

10% shareholders (Unlocks in stages)

5% Marketing and Develop Wallet (Not Locked)

Supply at Launch

25% Locked LP: 12 BNB for 25T Pancakeswap

25% to be added to Pancake swap on completion of stage one marketing (Not Locked as Yet for further dev)

Burn TX Click Below

Binance Transaction Hash (Txhash) Details | BscScan

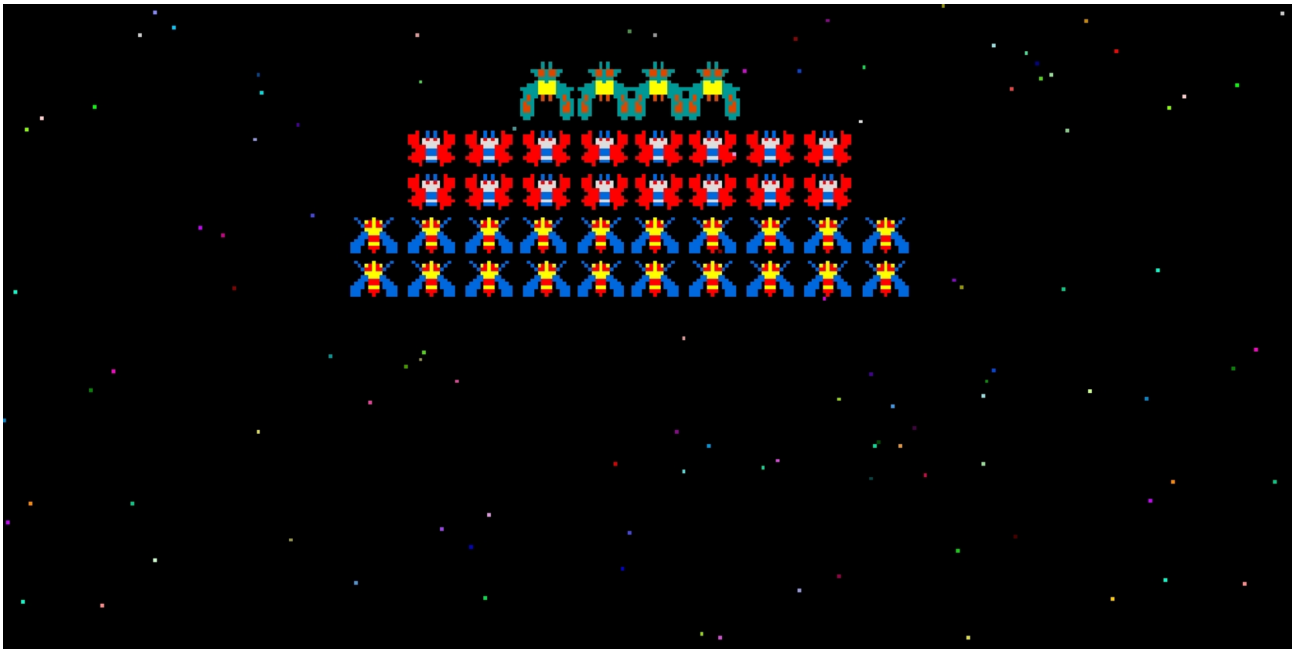
https://bscscan.com/tx/0x29a196d64c7fdfed5c0abdff050b7c8030464daebb824bc8b2add8caf2a8ea22

Airdrops

Still a surprise

10% of Supply to be Airdropped

join our telegram to find out more on how to qualify for airdrop



Our Links

Galaga Token	https://t.me/galagatoken
galagatoken/galagatoken	https://github.com/galagatoken/galagatoken
Cathie alt coin Erc-20 information	https://www.facebook.com/cathiecoinpage
galagatoken	https://twitter.com/galagatoken

Contact Us

Best way to get in contact with us is telegram we also have twitter and email

galagatokem@gmail.com

Galaga Token	https://t.me/galagatoken
Home | Galaga Token	https://www.galagatoken.com
galagatoken	https://twitter.com/galagatoken

How To Buy Galaga Token $GLG

How to Trade Galaga Token $GLG

Trading on PancakeSwap is very easy compared to most exchanges. You aren't going to be overwhelmed by charts or jargon, and calculations are all handled for you.

Getting set up to trade

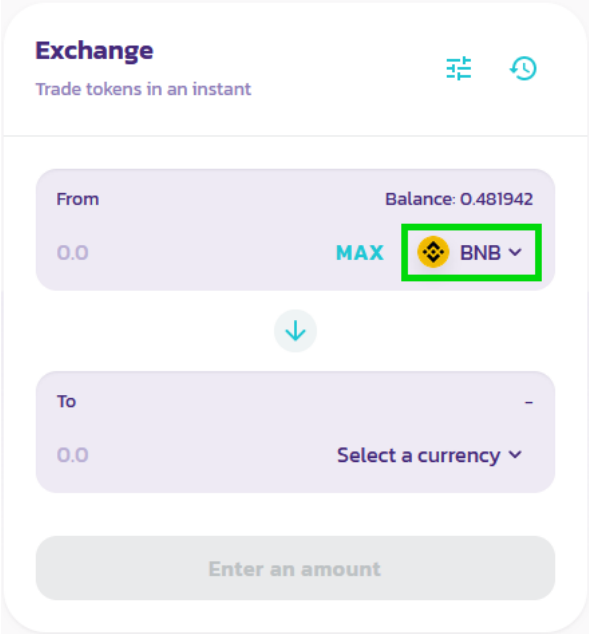
Before you can trade, you will need a Binance Smart Chain-compatible wallet. You can learn how to get one here. You will also need to have some BEP20 tokens to trade with. You can learn how to get some here.

Trading on the PancakeSwap exchange

1. Go to the exchange page here.

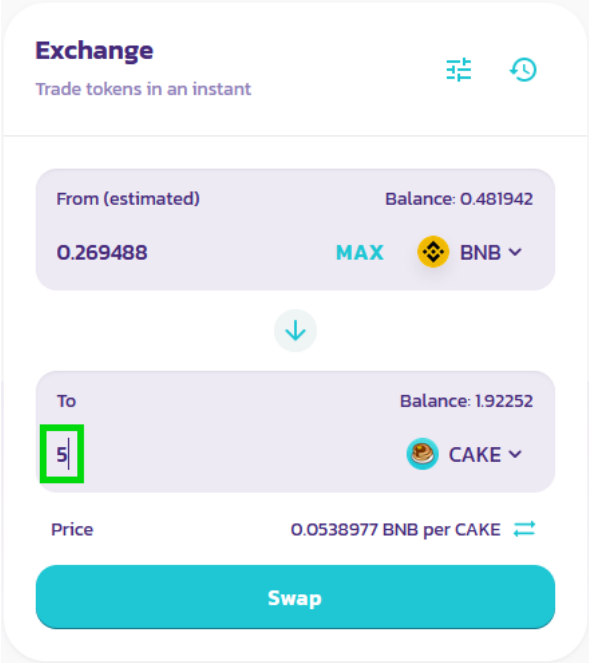
2. Unlock your Binance Smart Chain-compatible wallet by clicking **Unlock Wallet** (you can also **Connect** in the top right-hand corner). If you haven't yet connected your wallet to PancakeSwap, you can view the guide to here. Unlock Wallet

3. Choose the token you want to trade from the dropdown menu in the "From" section. The default setting is BNB.



Whichever token you choose, you will need to make sure you have some to trade with. Your balance is shown above the token dropdown menu.

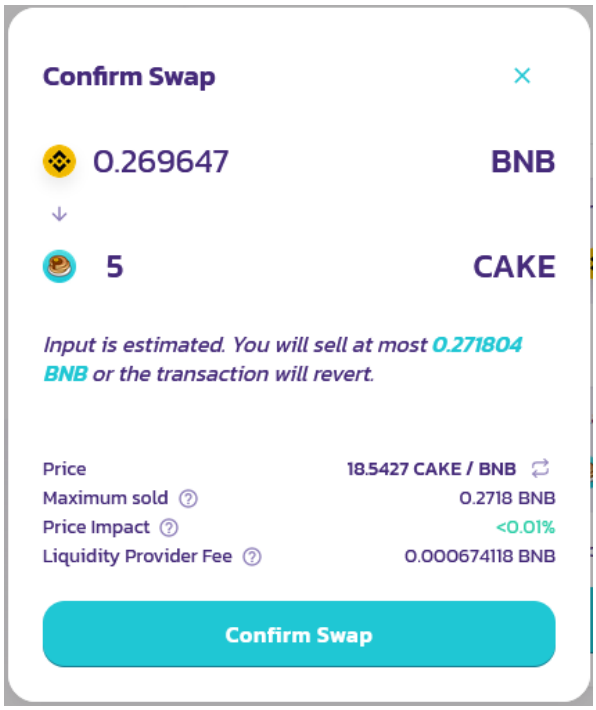
4. Choose the token you want to trade to in the "To" section as above. Next, type an amount for your "To" currency by clicking inside the input box.



Your "From" currency amount will be estimated automatically. You can also type your "From" amount and have the "To" amount estimate automatically if you like.

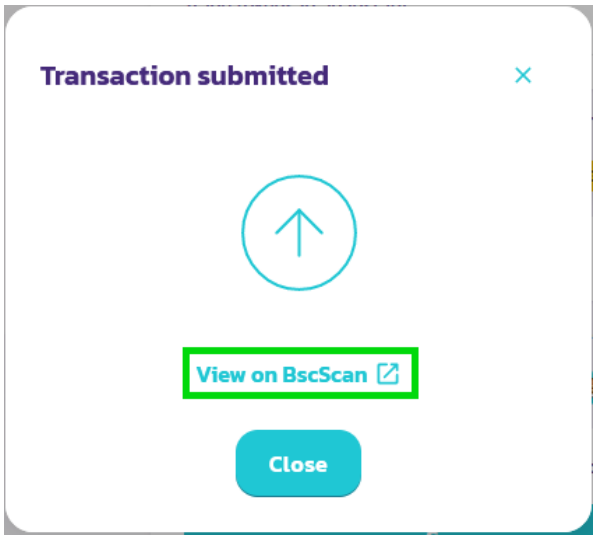
5. Check the details, and click the **Swap** button.

6. A window with more details will appear. Check the details are correct.



When you are ready, click the **Confirm Swap** button. Your wallet will ask you to confirm the action.

7. Done! You can click **View on BscScan** to see your transaction details on the explorer.



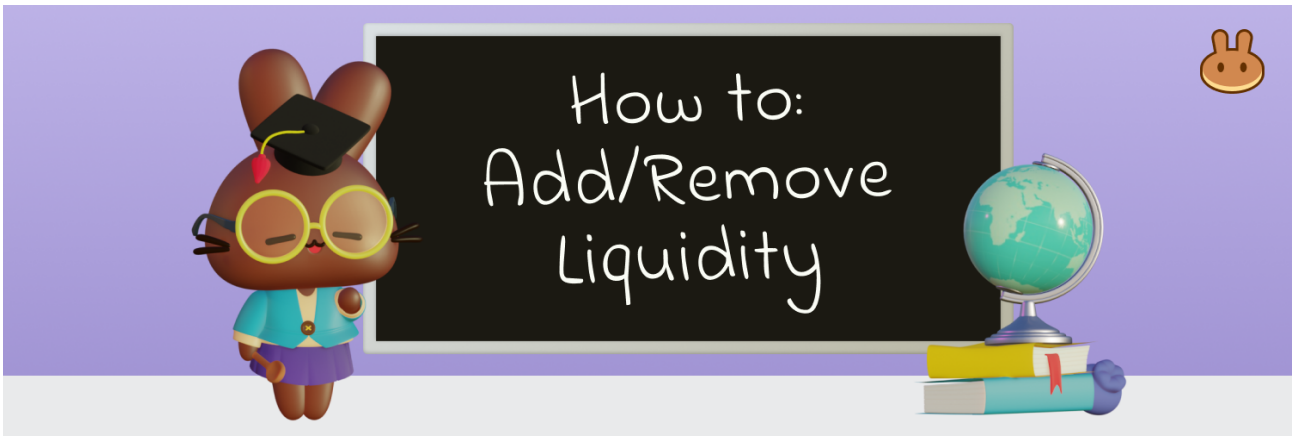
Connect to BSC to Buy Galagatoken

Use Metamask

Use Trustwallet

Add Galagatoken LP on Pancake Swap

How to Add/Remove Liquidity



"Liquidity" is central to how PancakeSwap's Exchange works. You can add liquidity for any token pair by staking both through the Liquidity page.

In return for adding liquidity, you'll receive trading fees for that pair, and receive LP Tokens you can stake in Farms to earn CAKE rewards!

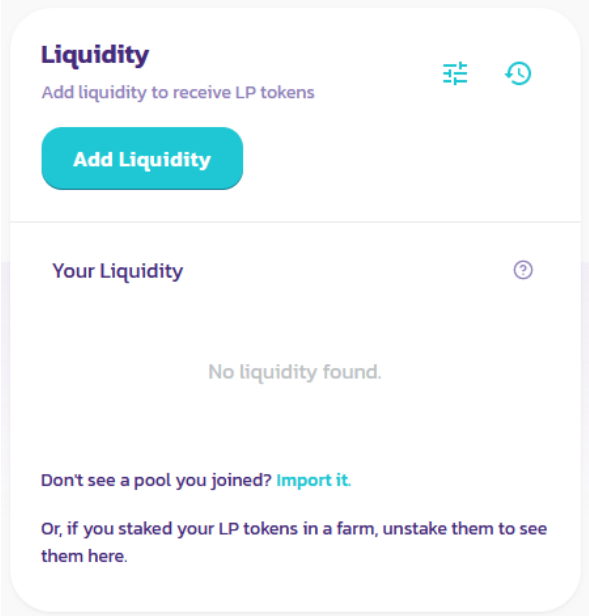
Adding liquidity

To provide liquidity, you'll need to commit an amount of any token pair you like. Your lowest value (in USD) of the two tokens will be the limit to the liquidity you can provide.

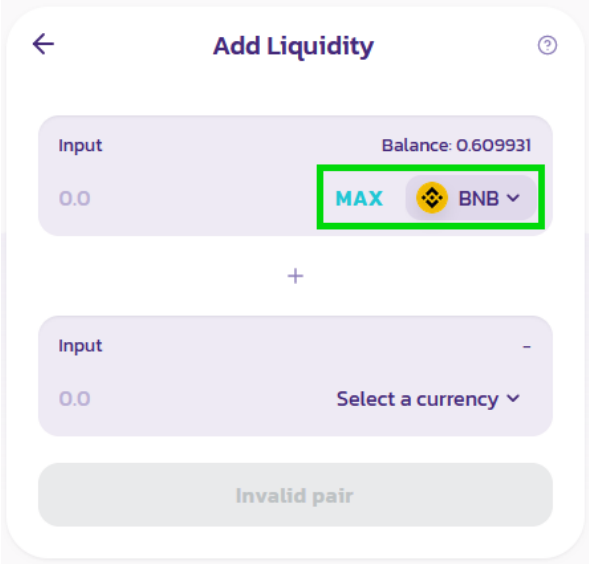
You can easily trade for any tokens you need. Visit our How to Trade on PancakeSwap guide if you need to.

In this example, we will add liquidity using BNB and CAKE.

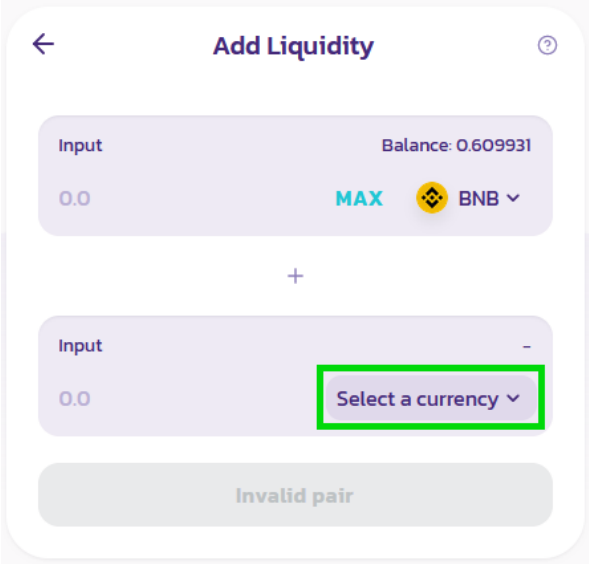
1. Visit the Liquidity page.



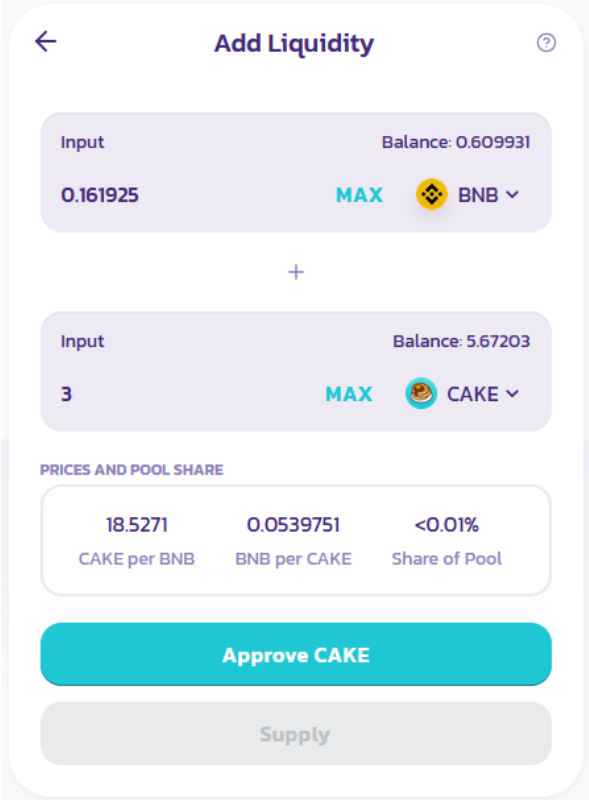
2. Click the **Add Liquidity** button.



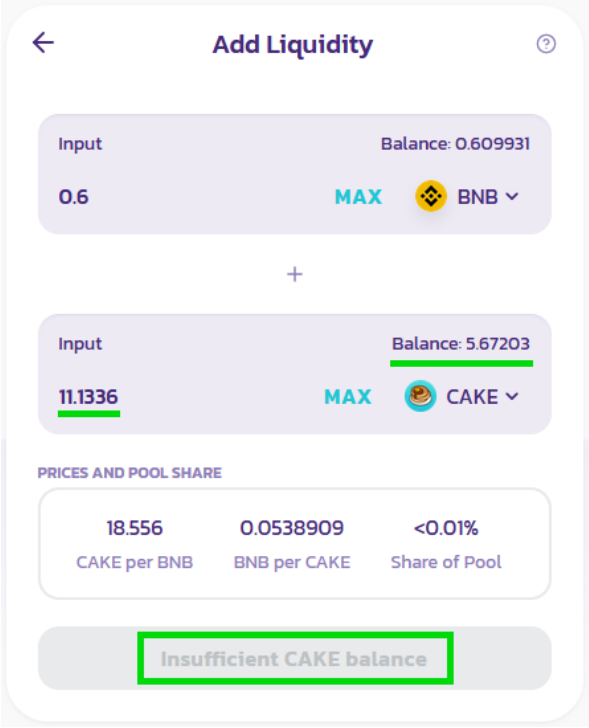
3. For the top Input, leave BNB as it is.



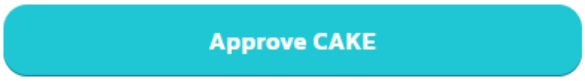
4. For the bottom input, click ‘Select a currency’ and pick CAKE.



5. Enter an amount on one of the tokens under “Input”. The other will calculate automatically.

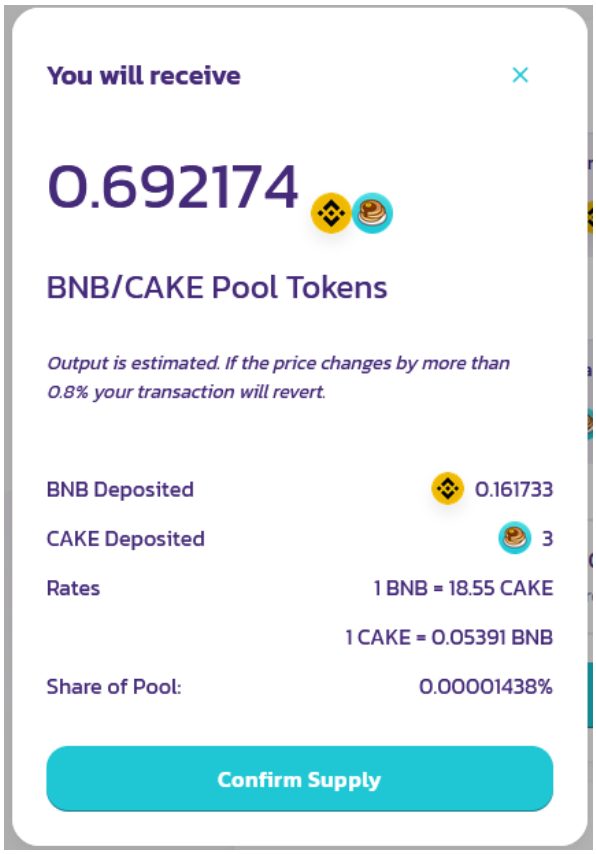


(If your balance is too low on one pair enter a lower amount.)

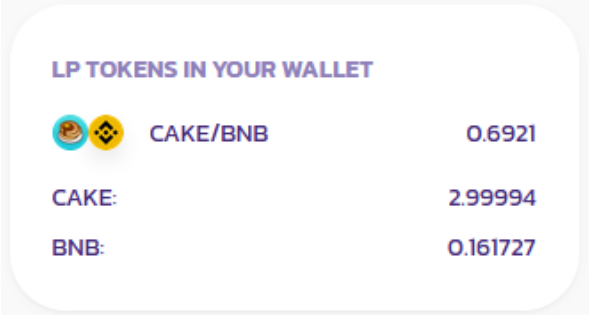


6. Click the **Approve CAKE** button. Your wallet will ask you to confirm the action.

7. The **Supply** button will light up. Click it.



8. A window will appear saying how much you will receive. Click the **Confirm Supply** button. Your wallet will ask you to confirm the action.



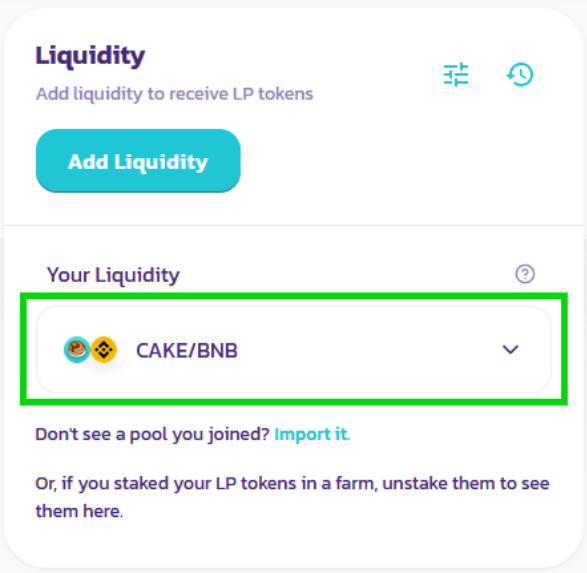
9. After a short wait you will see your LP Token balance at the bottom of the page.

You can repeat the steps above to add more liquidity at any time.

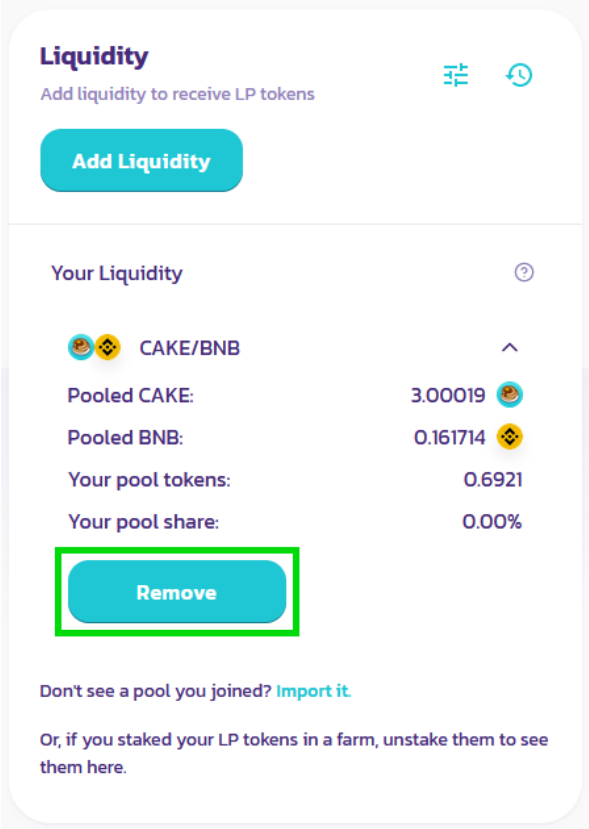
Removing liquidity

To remove liquidity.

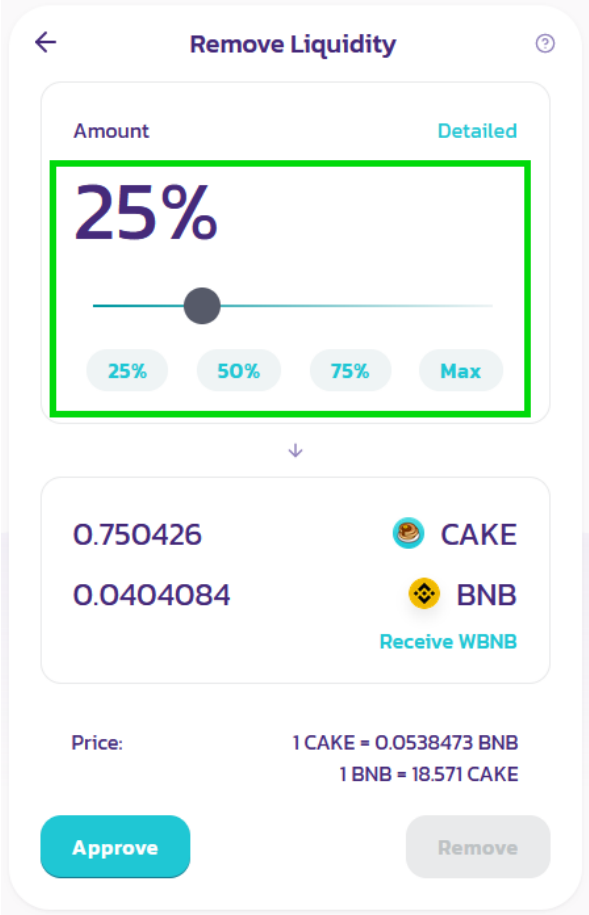
1. Visit the Liquidity page.



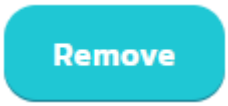
2. Click on your pair under “Your Liquidity”.



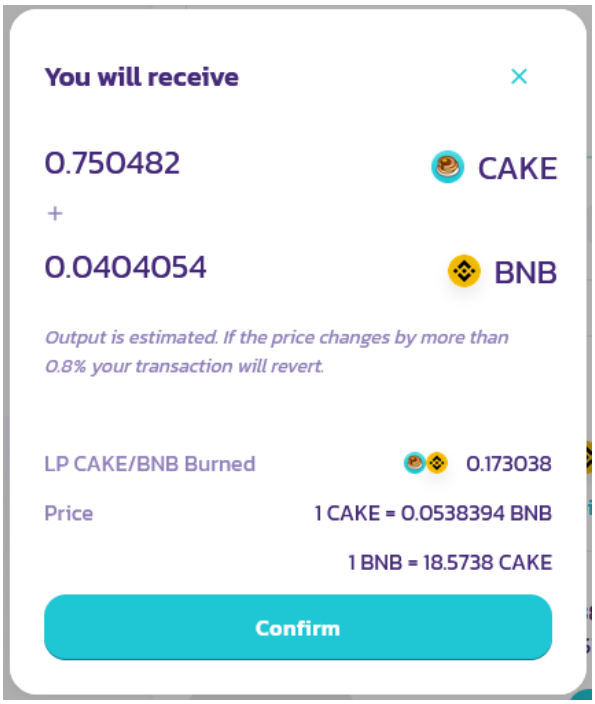
3. Click **Remove**. A new window will appear.



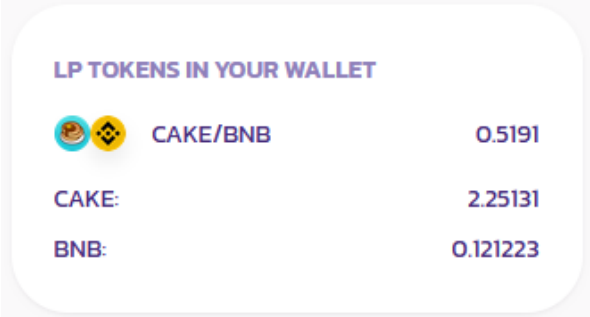
4. Use the buttons or slider to choose what percent to remove. Choose **MAX** to remove everything. 

5. Click **Approve**. Your wallet will ask you to confirm the action**.** 

6. The **Remove** button will light up. Click it.



7. A window will appear saying what you will receive. Click **Confirm**. Your wallet will ask you to confirm the action.



8. After a short wait you will see your new LP Token balance at the bottom of the page.

Contract

See below our Contract Commit to the Binance Smart Chain (BSC)

available also on our Github Page or bscscan **

```
/**
 *Submitted for verification at BscScan.com on 2021-06-06
*/

// SPDX-License-Identifier: Unlicensed

pragma solidity ^0.8.4;

interface IERC20 {

    function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns (uint256);
    function transfer(address recipient, uint256 amount) external returns
    function allowance(address owner, address spender) external view retur
    function approve(address spender, uint256 amount) external returns (bo
    function transferFrom(address sender, address recipient, uint256 amoun

    event Transfer(address indexed from, address indexed to, uint256 value
    event Approval(address indexed owner, address indexed spender, uint256
}

library SafeMath {

    function tryAdd(uint256 a, uint256 b) internal pure returns (bool, uin
        unchecked {
            uint256 c = a + b;
            if (c < a) return (false, 0);
            return (true, c);
        }
    }

    function trySub(uint256 a, uint256 b) internal pure returns (bool, uin
        unchecked {
            if (b > a) return (false, 0);
            return (true, a - b);
        }
    }

    function tryMul(uint256 a, uint256 b) internal pure returns (bool, uin
        unchecked {
```

```
            // Gas optimization: this is cheaper than requiring 'a' not be
            // benefit is lost if 'b' is also tested.
            // See: https://github.com/OpenZeppelin/openzeppelin-contracts
            if (a == 0) return (true, 0);
            uint256 c = a * b;
            if (c / a != b) return (false, 0);
            return (true, c);
        }
    }

    function tryDiv(uint256 a, uint256 b) internal pure returns (bool, uin
        unchecked {
            if (b == 0) return (false, 0);
            return (true, a / b);
        }
    }

    function tryMod(uint256 a, uint256 b) internal pure returns (bool, uin
        unchecked {
            if (b == 0) return (false, 0);
            return (true, a % b);
        }
    }

    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        return a + b;
    }


    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        return a - b;
    }


    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        return a * b;
    }

    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        return a / b;
    }


    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        return a % b;
    }

    function sub(uint256 a, uint256 b, string memory errorMessage) interna
        unchecked {
            require(b <= a, errorMessage);
            return a - b;
```

```
        }
    }

    function div(uint256 a, uint256 b, string memory errorMessage) interna
        unchecked {
            require(b > 0, errorMessage);
            return a / b;
        }
    }

    function mod(uint256 a, uint256 b, string memory errorMessage) interna
        unchecked {
            require(b > 0, errorMessage);
            return a % b;
        }
    }
}




abstract contract Context {
    function _msgSender() internal view virtual returns (address) {
        return msg.sender;
    }

    function _msgData() internal view virtual returns (bytes calldata) {
        this; // silence state mutability warning without generating bytec
        return msg.data;
    }
}


library Address {

    function isContract(address account) internal view returns (bool) {
        uint256 size;
        assembly { size := extcodesize(account) }
        return size > 0;
    }

    function sendValue(address payable recipient, uint256 amount) internal
        require(address(this).balance >= amount, "Address: insufficient ba
        (bool success, ) = recipient.call{ value: amount }("");
        require(success, "Address: unable to send value, recipient may hav
    }

    function functionCall(address target, bytes memory data) internal retu
      return functionCall(target, data, "Address: low-level call failed");
    }

```

```
    function functionCall(address target, bytes memory data, string memory
        return functionCallWithValue(target, data, 0, errorMessage);
    }

    function functionCallWithValue(address target, bytes memory data, uint
        return functionCallWithValue(target, data, value, "Address: low-le
    }

    function functionCallWithValue(address target, bytes memory data, uint
        require(address(this).balance >= value, "Address: insufficient bal
        require(isContract(target), "Address: call to non-contract");
        (bool success, bytes memory returndata) = target.call{ value: valu
        return _verifyCallResult(success, returndata, errorMessage);
    }

    function functionStaticCall(address target, bytes memory data) interna
        return functionStaticCall(target, data, "Address: low-level static
    }

    function functionStaticCall(address target, bytes memory data, string
        require(isContract(target), "Address: static call to non-contract"
        (bool success, bytes memory returndata) = target.staticcall(data);
        return _verifyCallResult(success, returndata, errorMessage);
    }


    function functionDelegateCall(address target, bytes memory data) inter
        return functionDelegateCall(target, data, "Address: low-level dele
    }

    function functionDelegateCall(address target, bytes memory data, strin
        require(isContract(target), "Address: delegate call to non-contrac
        (bool success, bytes memory returndata) = target.delegatecall(data
        return _verifyCallResult(success, returndata, errorMessage);
    }

    function _verifyCallResult(bool success, bytes memory returndata, stri
        if (success) {
            return returndata;
        } else {
            if (returndata.length > 0) {
                 assembly {
                    let returndata_size := mload(returndata)
                    revert(add(32, returndata), returndata_size)
                }
            } else {
                revert(errorMessage);
            }
        }
    }
}
```

```
194
195
196
197 abstract contract Ownable is Context {
198     address public _owner;
199     address private _previousOwner;
200     uint256 public _lockTime;
201
202     event OwnershipTransferred(address indexed previousOwner, address inde
203     constructor () {
204         _owner = _msgSender();
205         emit OwnershipTransferred(address(0), _owner);
206     }
207
208     function owner() public view virtual returns (address) {
209         return _owner;
210     }
211
212     modifier onlyOwner() {
213         require(owner() == _msgSender(), "Ownable: caller is not the owner
214         _;
215     }
216
217     function renounceOwnership() public virtual onlyOwner {
218         emit OwnershipTransferred(_owner, address(0));
219         _owner = address(0);
220     }
221
222
223     function transferOwnership(address newOwner) public virtual onlyOwner
224         require(newOwner != address(0), "Ownable: new owner is the zero ad
225         emit OwnershipTransferred(_owner, newOwner);
226         _owner = newOwner;
227     }
228
229
230         //Locks the contract for owner for the amount of time provided
231     function lock(uint256 time) public virtual onlyOwner {
232         _previousOwner = _owner;
233         _owner = address(0);
234         _lockTime = time;
235         emit OwnershipTransferred(_owner, address(0));
236     }
237
238     //Unlocks the contract for owner when _lockTime is exceeds
239     function unlock() public virtual {
240         require(_previousOwner == msg.sender, "You don't have permission t
241         require(block.timestamp > _lockTime , "Contract is locked.");
242         emit OwnershipTransferred(_owner, _previousOwner);
243         _owner = _previousOwner;
244     }
```

```
245 }
246
247 interface IUniswapV2Factory {
248     event PairCreated(address indexed token0, address indexed token1, addr
249     function feeTo() external view returns (address);
250     function feeToSetter() external view returns (address);
251     function getPair(address tokenA, address tokenB) external view returns
252     function allPairs(uint) external view returns (address pair);
253     function allPairsLength() external view returns (uint);
254     function createPair(address tokenA, address tokenB) external returns (
255     function setFeeTo(address) external;
256     function setFeeToSetter(address) external;
257 }
258
259 interface IUniswapV2Pair {
260     event Approval(address indexed owner, address indexed spender, uint va
261     event Transfer(address indexed from, address indexed to, uint value);
262     function name() external pure returns (string memory);
263     function symbol() external pure returns (string memory);
264     function decimals() external pure returns (uint8);
265     function totalSupply() external view returns (uint);
266     function balanceOf(address owner) external view returns (uint);
267     function allowance(address owner, address spender) external view retur
268     function approve(address spender, uint value) external returns (bool);
269     function transfer(address to, uint value) external returns (bool);
270     function transferFrom(address from, address to, uint value) external r
271     function DOMAIN_SEPARATOR() external view returns (bytes32);
272     function PERMIT_TYPEHASH() external pure returns (bytes32);
273     function nonces(address owner) external view returns (uint);
274     function permit(address owner, address spender, uint value, uint deadl
275     event Mint(address indexed sender, uint amount0, uint amount1);
276     event Burn(address indexed sender, uint amount0, uint amount1, address
277     event Swap(
278         address indexed sender,
279         uint amount0In,
280         uint amount1In,
281         uint amount0Out,
282         uint amount1Out,
283         address indexed to
284     );
285     event Sync(uint112 reserve0, uint112 reserve1);
286     function MINIMUM_LIQUIDITY() external pure returns (uint);
287     function factory() external view returns (address);
288     function token0() external view returns (address);
289     function token1() external view returns (address);
290     function getReserves() external view returns (uint112 reserve0, uint11
291     function price0CumulativeLast() external view returns (uint);
292     function price1CumulativeLast() external view returns (uint);
293     function kLast() external view returns (uint);
294     function mint(address to) external returns (uint liquidity);
295     function burn(address to) external returns (uint amount0, uint amount1
```

```
    function swap(uint amount0Out, uint amount1Out, address to, bytes call
    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 liq
    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 p
        external
        payable
        returns (uint[] memory amounts);
    function swapTokensForExactETH(uint amountOut, uint amountInMax, addre
        external
        returns (uint[] memory amounts);
    function swapExactTokensForETH(uint amountIn, uint amountOutMin, addre
        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 pu
    function getAmountOut(uint amountIn, uint reserveIn, uint reserveOut)
    function getAmountIn(uint amountOut, uint reserveIn, uint reserveOut)
    function getAmountsOut(uint amountIn, address[] calldata path) externa
    function getAmountsIn(uint amountOut, address[] calldata path) externa
}

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;
}

contract CoinToken is Context, IERC20, Ownable {
    using SafeMath for uint256;
    using Address for address;

    mapping (address => uint256) private _rOwned;
    mapping (address => uint256) private _tOwned;
    mapping (address => mapping (address => uint256)) private _allowances;
    mapping (address => bool) private _isExcludedFromFee;
    mapping (address => bool) private _isExcluded;
    address[] private _excluded;
    address public _devWalletAddress;     // TODO - team wallet here
    uint256 private constant MAX = ~uint256(0);
    uint256 private _tTotal;
```

```
    uint256 private _rTotal;
    uint256 private _tFeeTotal;
    string private _name;
    string private _symbol;
    uint256 private _decimals;
    uint256 public _taxFee;
    uint256 private _previousTaxFee;
    uint256 public _devFee;
    uint256 private _previousDevFee;
    uint256 public _liquidityFee;
    uint256 private _previousLiquidityFee;
    IUniswapV2Router02 public uniswapV2Router;
    address public uniswapV2Pair;
    bool inSwapAndLiquify;
    bool public swapAndLiquifyEnabled = true;
    uint256 public _maxTxAmount;
    uint256 public numTokensSellToAddToLiquidity;
    event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
    event SwapAndLiquifyEnabledUpdated(bool enabled);
    event SwapAndLiquify(
        uint256 tokensSwapped,
        uint256 ethReceived,
        uint256 tokensIntoLiqudity
    );

    modifier lockTheSwap {
        inSwapAndLiquify = true;
        _;
        inSwapAndLiquify = false;
    }

    constructor (string memory _NAME, string memory _SYMBOL, uint256 _DECI
        _name = _NAME;
        _symbol = _SYMBOL;
        _decimals = _DECIMALS;
        _tTotal = _supply * 10 ** _decimals;
        _rTotal = (MAX - (MAX % _tTotal));
        _taxFee = _txFee;
        _liquidityFee = _lpFee;
        _previousTaxFee = _txFee;

    _devFee = _DexFee;
    _previousDevFee = _devFee;
        _previousLiquidityFee = _lpFee;
        _maxTxAmount = (_tTotal * 5 / 1000) * 10 ** _decimals;
        numTokensSellToAddToLiquidity = (_tTotal * 5 / 10000) * 10 ** _dec
        _devWalletAddress = feeaddress;

        _rOwned[tokenOwner] = _rTotal;

        IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(routerAdd
```

```
500          // Create a uniswap pair for this new token
501         uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
502             .createPair(address(this), _uniswapV2Router.WETH());
503
504         // set the rest of the contract variables
505         uniswapV2Router = _uniswapV2Router;
506
507         //exclude owner and this contract from fee
508         _isExcludedFromFee[tokenOwner] = true;
509         _isExcludedFromFee[address(this)] = true;
510
511         _owner = tokenOwner;
512         payable(service).transfer(msg.value);
513         emit Transfer(address(0), tokenOwner, _tTotal);
514
515
516     }
517
518     function name() public view returns (string memory) {
519         return _name;
520     }
521
522     function symbol() public view returns (string memory) {
523         return _symbol;
524     }
525
526     function decimals() public view returns (uint256) {
527         return _decimals;
528     }
529
530     function totalSupply() public view override returns (uint256) {
531         return _tTotal;
532     }
533
534     function balanceOf(address account) public view override returns (uint
535         if (_isExcluded[account]) return _tOwned[account];
536         return tokenFromReflection(_rOwned[account]);
537     }
538
539     function transfer(address recipient, uint256 amount) public override r
540         _transfer(_msgSender(), recipient, amount);
541         return true;
542     }
543
544     function allowance(address owner, address spender) public view overrid
545         return _allowances[owner][spender];
546     }
547
548     function approve(address spender, uint256 amount) public override retu
549         _approve(_msgSender(), spender, amount);
550         return true;
```

```
    }

    function transferFrom(address sender, address recipient, uint256 amoun
        _transfer(sender, recipient, amount);
        _approve(sender, _msgSender(), _allowances[sender][_msgSender()].s
        return true;
    }

    function increaseAllowance(address spender, uint256 addedValue) public
        _approve(_msgSender(), spender, _allowances[_msgSender()][spender]
        return true;
    }

    function decreaseAllowance(address spender, uint256 subtractedValue) p
        _approve(_msgSender(), spender, _allowances[_msgSender()][spender]
        return true;
    }

    function isExcludedFromReward(address account) public view returns (bo
        return _isExcluded[account];
    }

    function totalFees() public view returns (uint256) {
        return _tFeeTotal;
    }

    function deliver(uint256 tAmount) public {
        address sender = _msgSender();
        require(!_isExcluded[sender], "Excluded addresses cannot call this
        (uint256 rAmount,,,,,,) = _getValues(tAmount);
        _rOwned[sender] = _rOwned[sender].sub(rAmount);
        _rTotal = _rTotal.sub(rAmount);
        _tFeeTotal = _tFeeTotal.add(tAmount);
    }

    function reflectionFromToken(uint256 tAmount, bool deductTransferFee)
        require(tAmount <= _tTotal, "Amount must be less than supply");
        if (!deductTransferFee) {
            (uint256 rAmount,,,,,,) = _getValues(tAmount);
            return rAmount;
        } else {
            (,uint256 rTransferAmount,,,,,) = _getValues(tAmount);
            return rTransferAmount;
        }
    }

    function tokenFromReflection(uint256 rAmount) public view returns(uint
        require(rAmount <= _rTotal, "Amount must be less than total reflec
        uint256 currentRate =  _getRate();
        return rAmount.div(currentRate);
    }
```

```
    function excludeFromReward(address account) public onlyOwner() {
        require(!_isExcluded[account], "Account is already excluded");
        if(_rOwned[account] > 0) {
            _tOwned[account] = tokenFromReflection(_rOwned[account]);
        }
        _isExcluded[account] = true;
        _excluded.push(account);
    }

    function includeInReward(address account) external onlyOwner() {
        require(_isExcluded[account], "Account is already included");
        for (uint256 i = 0; i < _excluded.length; i++) {
            if (_excluded[i] == account) {
                _excluded[i] = _excluded[_excluded.length - 1];
                _tOwned[account] = 0;
                _isExcluded[account] = false;
                _excluded.pop();
                break;
            }
        }
    }
        function _transferBothExcluded(address sender, address recipient,
        (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 t
        _tOwned[sender] = _tOwned[sender].sub(tAmount);
        _rOwned[sender] = _rOwned[sender].sub(rAmount);
        _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount);
        _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
        _takeLiquidity(tLiquidity);
        _takeDev(tDev);
        _reflectFee(rFee, tFee);
        emit Transfer(sender, recipient, tTransferAmount);
    }

    function excludeFromFee(address account) public onlyOwner {
        _isExcludedFromFee[account] = true;
    }

    function includeInFee(address account) public onlyOwner {
        _isExcludedFromFee[account] = false;
    }

    function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
        _taxFee = taxFee;
    }

    function setDevFeePercent(uint256 devFee) external onlyOwner() {
        _devFee = devFee;
    }

    function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwn
```

```
        _liquidityFee = liquidityFee;
    }

    function setMaxTxPercent(uint256 maxTxPercent) public onlyOwner {
        _maxTxAmount = maxTxPercent  * 10 ** _decimals;
    }

    function setDevWalletAddress(address _addr) public onlyOwner {
        _devWalletAddress = _addr;
    }


    function setSwapAndLiquifyEnabled(bool _enabled) public onlyOwner {
        swapAndLiquifyEnabled = _enabled;
        emit SwapAndLiquifyEnabledUpdated(_enabled);
    }

     //to recieve ETH from uniswapV2Router when swaping
    receive() external payable {}

    function _reflectFee(uint256 rFee, uint256 tFee) private {
        _rTotal = _rTotal.sub(rFee);
        _tFeeTotal = _tFeeTotal.add(tFee);
    }

    function _getValues(uint256 tAmount) private view returns (uint256, ui
        (uint256 tTransferAmount, uint256 tFee, uint256 tLiquidity, uint25
        (uint256 rAmount, uint256 rTransferAmount, uint256 rFee) = _getRVa
        return (rAmount, rTransferAmount, rFee, tTransferAmount, tFee, tLi
    }

    function _getTValues(uint256 tAmount) private view returns (uint256, u
        uint256 tFee = calculateTaxFee(tAmount);
        uint256 tLiquidity = calculateLiquidityFee(tAmount);
        uint256 tDev = calculateDevFee(tAmount);
        uint256 tTransferAmount = tAmount.sub(tFee).sub(tLiquidity).sub(tD
        return (tTransferAmount, tFee, tLiquidity, tDev);
    }

    function _getRValues(uint256 tAmount, uint256 tFee, uint256 tLiquidity
        uint256 rAmount = tAmount.mul(currentRate);
        uint256 rFee = tFee.mul(currentRate);
        uint256 rLiquidity = tLiquidity.mul(currentRate);
        uint256 rDev = tDev.mul(currentRate);
        uint256 rTransferAmount = rAmount.sub(rFee).sub(rLiquidity).sub(rD
        return (rAmount, rTransferAmount, rFee);
    }

    function _getRate() private view returns(uint256) {
        (uint256 rSupply, uint256 tSupply) = _getCurrentSupply();
        return rSupply.div(tSupply);
```

```
    }

    function _getCurrentSupply() private view returns(uint256, uint256) {
        uint256 rSupply = _rTotal;
        uint256 tSupply = _tTotal;
        for (uint256 i = 0; i < _excluded.length; i++) {
            if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] >
            rSupply = rSupply.sub(_rOwned[_excluded[i]]);
            tSupply = tSupply.sub(_tOwned[_excluded[i]]);
        }
        if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
        return (rSupply, tSupply);
    }

    function _takeLiquidity(uint256 tLiquidity) private {
        uint256 currentRate =  _getRate();
        uint256 rLiquidity = tLiquidity.mul(currentRate);
        _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity);
        if(_isExcluded[address(this)])
            _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity
    }

    function _takeDev(uint256 tDev) private {
        uint256 currentRate =  _getRate();
        uint256 rDev = tDev.mul(currentRate);
        _rOwned[_devWalletAddress] = _rOwned[_devWalletAddress].add(rDev);
        if(_isExcluded[_devWalletAddress])
            _tOwned[_devWalletAddress] = _tOwned[_devWalletAddress].add(tD
    }

    function calculateTaxFee(uint256 _amount) private view returns (uint25
        return _amount.mul(_taxFee).div(
            10**2
        );
    }

    function calculateDevFee(uint256 _amount) private view returns (uint25
        return _amount.mul(_devFee).div(
            10**2
        );
    }

    function calculateLiquidityFee(uint256 _amount) private view returns (
        return _amount.mul(_liquidityFee).div(
            10**2
        );
    }

    function removeAllFee() private {
        _previousTaxFee = _taxFee;
        _previousDevFee = _devFee;
```

```
        _previousLiquidityFee = _liquidityFee;

        _taxFee = 0;
        _devFee = 0;
        _liquidityFee = 0;
    }

    function restoreAllFee() private {
        _taxFee = _previousTaxFee;
        _devFee = _previousDevFee;
        _liquidityFee = _previousLiquidityFee;
    }

    function isExcludedFromFee(address account) public view returns(bool)
        return _isExcludedFromFee[account];
    }

    function _approve(address owner, address spender, uint256 amount) priv
        require(owner != address(0), "ERC20: approve from the zero address
        require(spender != address(0), "ERC20: approve to the zero address

        _allowances[owner][spender] = amount;
        emit Approval(owner, spender, amount);
    }

    function _transfer(
        address from,
        address to,
        uint256 amount
    ) private {
        require(from != address(0), "ERC20: transfer from the zero address
        require(to != address(0), "ERC20: transfer to the zero address");
        require(amount > 0, "Transfer amount must be greater than zero");
        if(from != owner() && to != owner())
            require(amount <= _maxTxAmount, "Transfer amount exceeds the m

        uint256 contractTokenBalance = balanceOf(address(this));

        if(contractTokenBalance >= _maxTxAmount)
        {
            contractTokenBalance = _maxTxAmount;
        }

        bool overMinTokenBalance = contractTokenBalance >= numTokensSellTo
        if (
            overMinTokenBalance &&
            !inSwapAndLiquify &&
            from != uniswapV2Pair &&
            swapAndLiquifyEnabled
        ) {
            contractTokenBalance = numTokensSellToAddToLiquidity;
```

```
            swapAndLiquify(contractTokenBalance);
        }

        bool takeFee = true;
        if(_isExcludedFromFee[from] || _isExcludedFromFee[to]){
            takeFee = false;
        }

        _tokenTransfer(from,to,amount,takeFee);
    }

    function swapAndLiquify(uint256 contractTokenBalance) private lockTheS
        uint256 half = contractTokenBalance.div(2);
        uint256 otherHalf = contractTokenBalance.sub(half);
        uint256 initialBalance = address(this).balance;
        swapTokensForEth(half);
        uint256 newBalance = address(this).balance.sub(initialBalance);
        addLiquidity(otherHalf, newBalance);
        emit SwapAndLiquify(half, newBalance, otherHalf);
    }

    function swapTokensForEth(uint256 tokenAmount) private {
        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = uniswapV2Router.WETH();
        _approve(address(this), address(uniswapV2Router), tokenAmount);
        uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens
            tokenAmount,
            0, // accept any amount of ETH
            path,
            address(this),
            block.timestamp
        );
    }

    function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private
        _approve(address(this), address(uniswapV2Router), tokenAmount);
        uniswapV2Router.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount,
            0, // slippage is unavoidable
            0, // slippage is unavoidable
            owner(),
            block.timestamp
        );
    }

    function _tokenTransfer(address sender, address recipient, uint256 amo
        if(!takeFee)
            removeAllFee();

```

```
        if (_isExcluded[sender] && !_isExcluded[recipient]) {
            _transferFromExcluded(sender, recipient, amount);
        } else if (!_isExcluded[sender] && _isExcluded[recipient]) {
            _transferToExcluded(sender, recipient, amount);
        } else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
            _transferStandard(sender, recipient, amount);
        } else if (_isExcluded[sender] && _isExcluded[recipient]) {
            _transferBothExcluded(sender, recipient, amount);
        } else {
            _transferStandard(sender, recipient, amount);
        }

        if(!takeFee)
            restoreAllFee();
    }

    function _transferStandard(address sender, address recipient, uint256
        (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 t
        _rOwned[sender] = _rOwned[sender].sub(rAmount);
        _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
        _takeLiquidity(tLiquidity);
        _takeDev(tDev);
        _reflectFee(rFee, tFee);
        emit Transfer(sender, recipient, tTransferAmount);
    }

    function _transferToExcluded(address sender, address recipient, uint25
        (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 t
        _rOwned[sender] = _rOwned[sender].sub(rAmount);
        _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount);
        _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
        _takeLiquidity(tLiquidity);
        _takeDev(tDev);
        _reflectFee(rFee, tFee);
        emit Transfer(sender, recipient, tTransferAmount);
    }

    function _transferFromExcluded(address sender, address recipient, uint
        (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 t
        _tOwned[sender] = _tOwned[sender].sub(tAmount);
        _rOwned[sender] = _rOwned[sender].sub(rAmount);
        _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
        _takeLiquidity(tLiquidity);
        _takeDev(tDev);
        _reflectFee(rFee, tFee);
        emit Transfer(sender, recipient, tTransferAmount);
    }


    function setRouterAddress(address newRouter) external onlyOwner {
        IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(newRouter
```

```
        uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).crea
        uniswapV2Router = _uniswapV2Router;
    }

    function setNumTokensSellToAddToLiquidity(uint256 amountToUpdate) exte
        numTokensSellToAddToLiquidity = amountToUpdate;
    }



}
```