

**Galaga Token \$GLG**

# Introduction

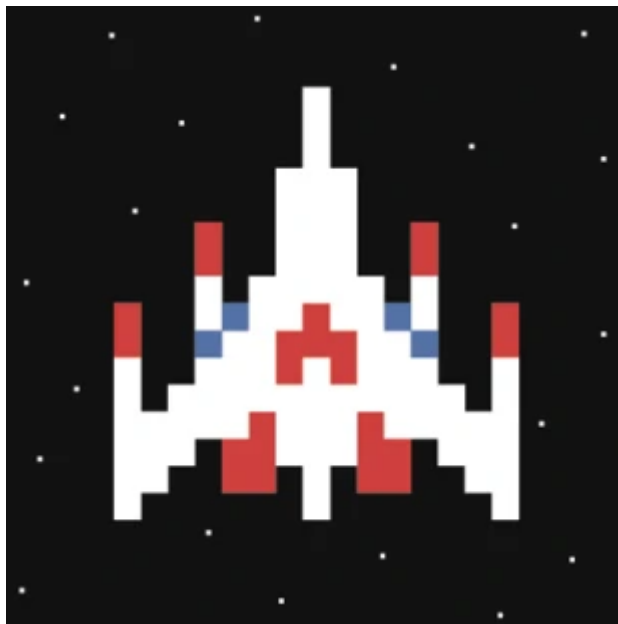
## About Us

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**

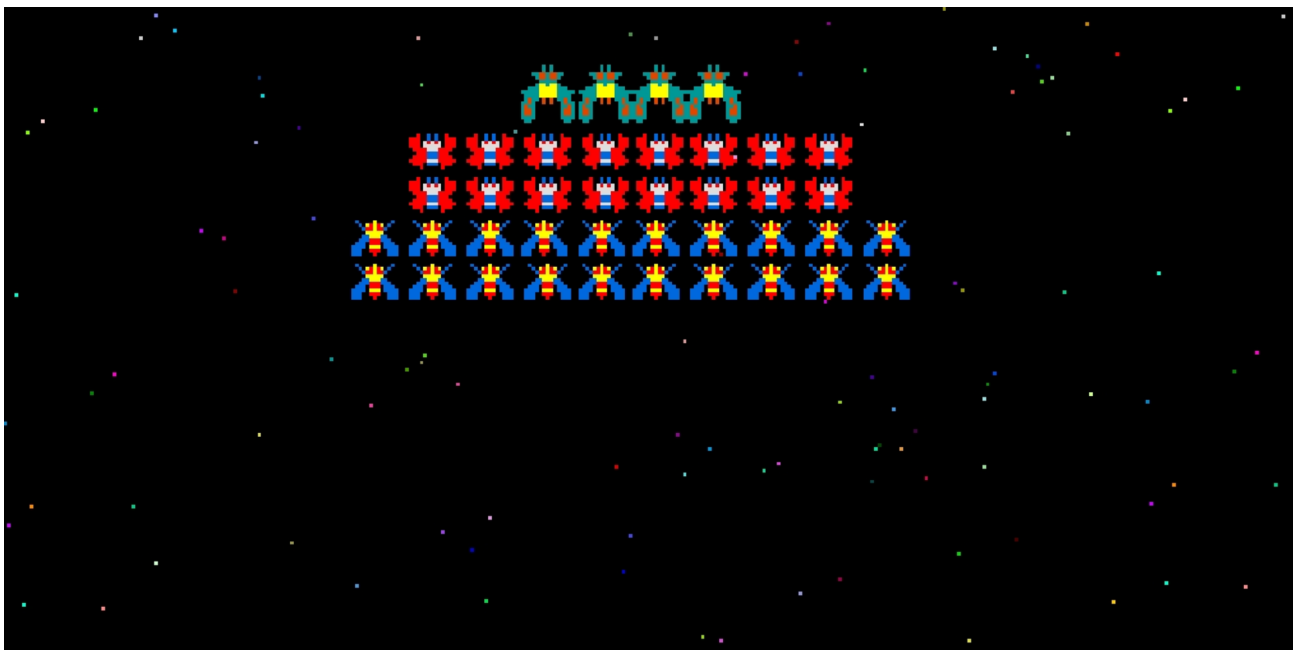
<https://bscscan.com/tx/0x29a196d64c7fdfed5c0abdff050b7c8030464daebb824bc8b2add8caf2a8ea22>

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

Unlock Wallet



The image shows the 'Exchange' interface with the title 'Exchange' and subtitle 'Trade tokens in an instant'. There are two icons in the top right: a list icon and a refresh icon. The 'From' section has a balance of 0.481942. The input field shows '0.0', a 'MAX' button, and a dropdown menu with the BNB token icon and 'BNB' selected. A green box highlights the BNB dropdown. Below the 'From' section is a downward arrow. The 'To' section shows '0.0' and a 'Select a currency' dropdown. At the bottom is a button labeled 'Enter an amount'.

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.

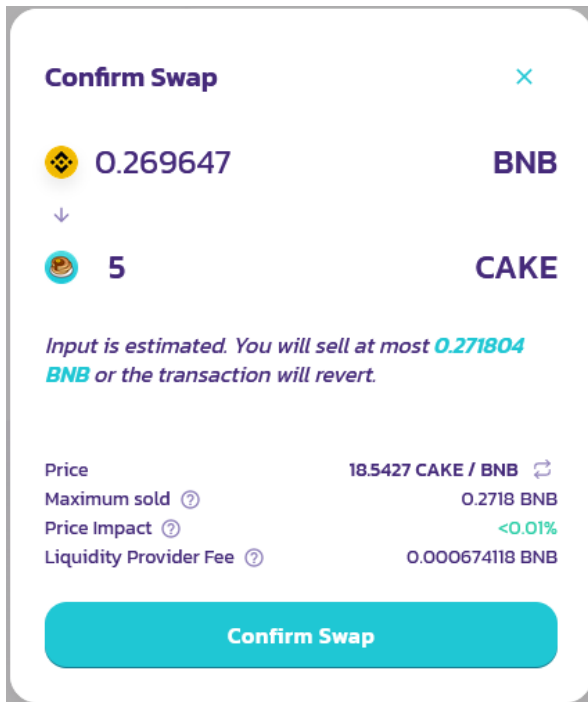
The image shows the 'Exchange' interface with the title 'Exchange' and subtitle 'Trade tokens in an instant'. There are two icons in the top right: a list icon and a refresh icon. The 'From (estimated)' section has a balance of 0.481942. The input field shows '0.269488', a 'MAX' button, and a dropdown menu with the BNB token icon and 'BNB' selected. Below the 'From' section is a downward arrow. The 'To' section has a balance of 1.92252. The input field shows '5' and a dropdown menu with the CAKE token icon and 'CAKE' selected. A green box highlights the '5' in the input field. Below the 'To' section is the 'Price' section showing '0.0538977 BNB per CAKE' with a double-headed arrow icon. At the bottom is a large blue button labeled 'Swap'.

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.

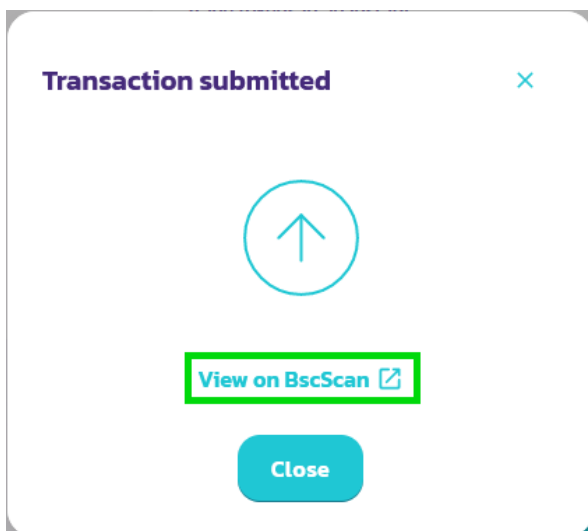
Swap

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.



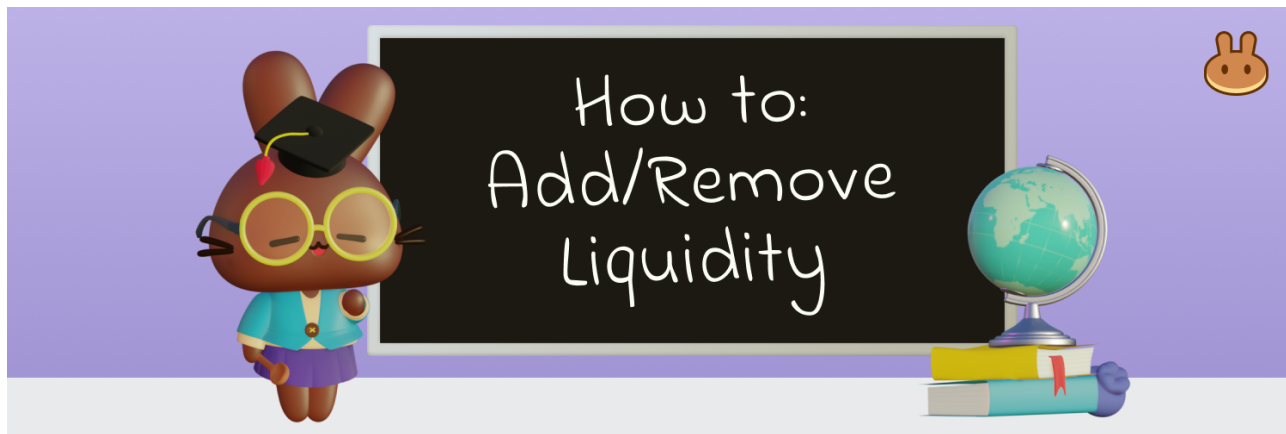
# Connect to BSC to Buy Galagatoken

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!

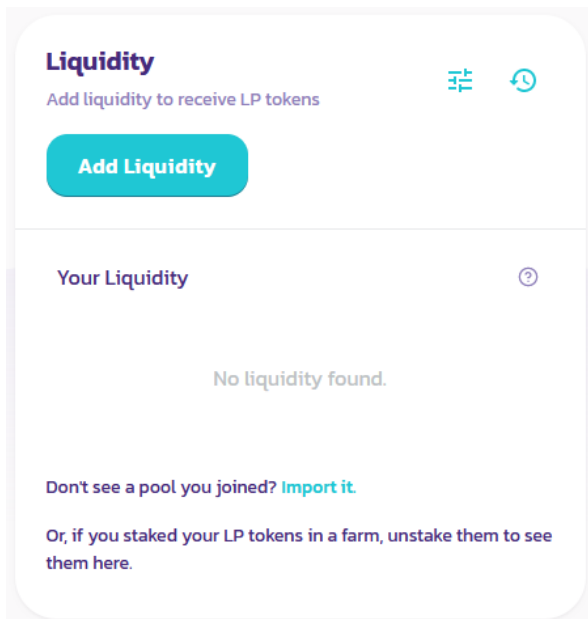
### Adding liquidity

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.

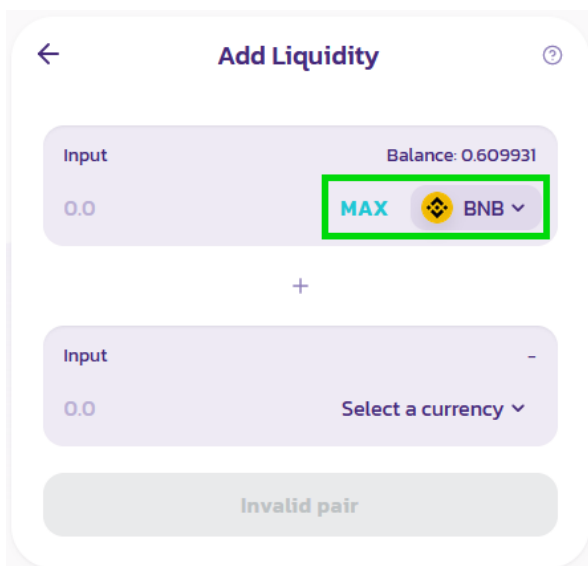
You can easily trade for any tokens you need. Visit our [How to Trade on PancakeSwap](#) guide if you need to.

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.

←

Add Liquidity


?

Input

Balance: 0.609931

0.0

MAX

 BNB ▾

+

Input

Balance: -

0.0

Select a currency ▾

Invalid pair

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

←

Add Liquidity


?

Input

Balance: 0.609931

0.161925

MAX

 BNB ▾


+

Input

Balance: 5.67203

3

MAX

 CAKE ▾

PRICES AND POOL SHARE

18.5271

0.0539751

<0.01%

CAKE per BNB

BNB per CAKE

Share of Pool

Approve CAKE

Supply

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

automatically.

The screenshot shows the 'Add Liquidity' screen. At the top, there's a back arrow and a help icon. Below, two input fields are shown with a plus sign between them. The first field is for BNB, with an input of 0.6 and a balance of 0.609931. The second field is for CAKE, with an input of 11.1336 and a balance of 5.67203. Below these is a section titled 'PRICES AND POOL SHARE' containing a table with three columns: 'CAKE per BNB', 'BNB per CAKE', and 'Share of Pool'. The values are 18.556, 0.0538909, and <0.01% respectively. At the bottom, a button labeled 'Insufficient CAKE balance' is highlighted with a green border.

PRICES AND POOL SHARE		
18.556	0.0538909	<0.01%
CAKE per BNB	BNB per CAKE	Share of Pool

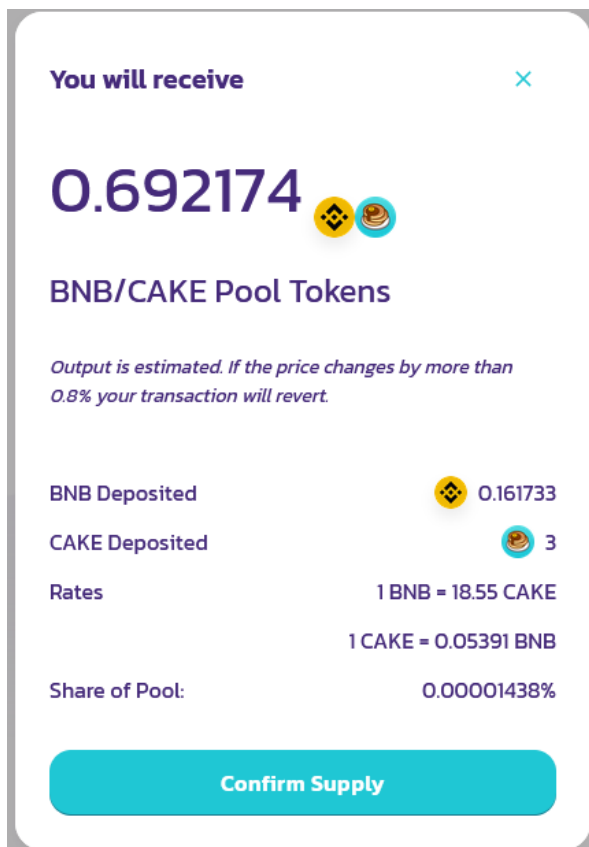
(If your balance is too low on one pair enter a lower amount.)

Approve CAKE

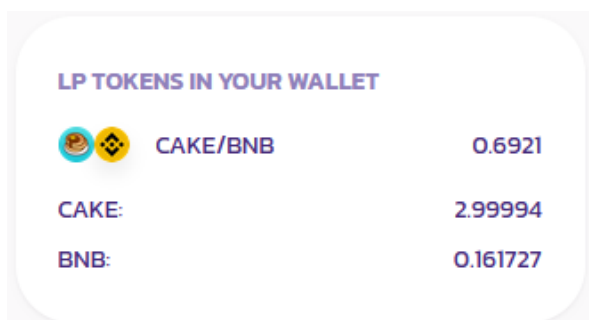
6. Click the **Approve CAKE** button. Your wallet will ask you to confirm the action.

Supply

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.

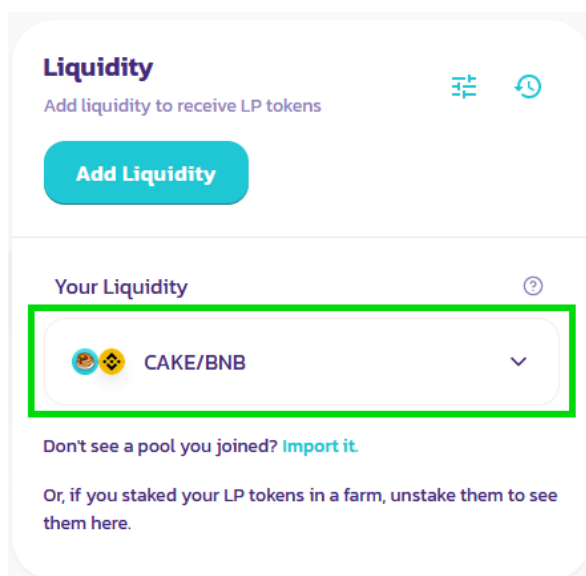
You can repeat the steps above to add more liquidity at any time.

## Removing liquidity

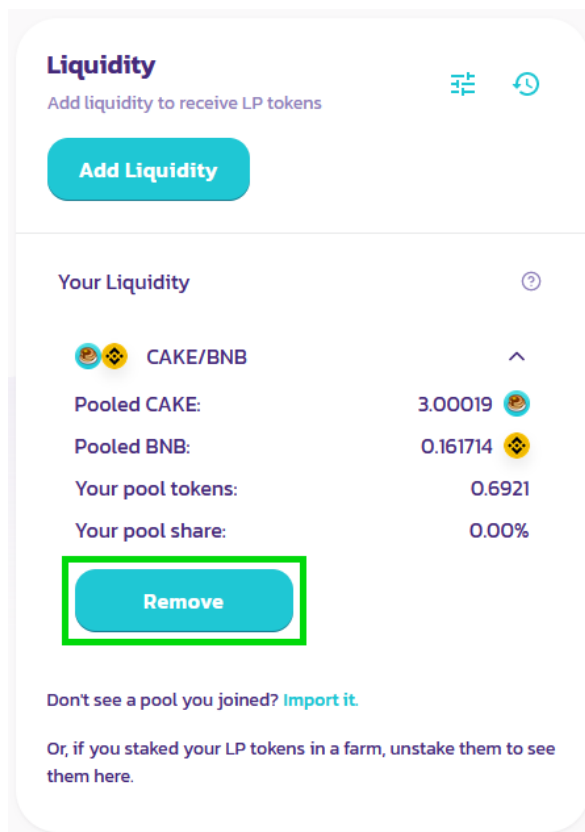
To remove liquidity.



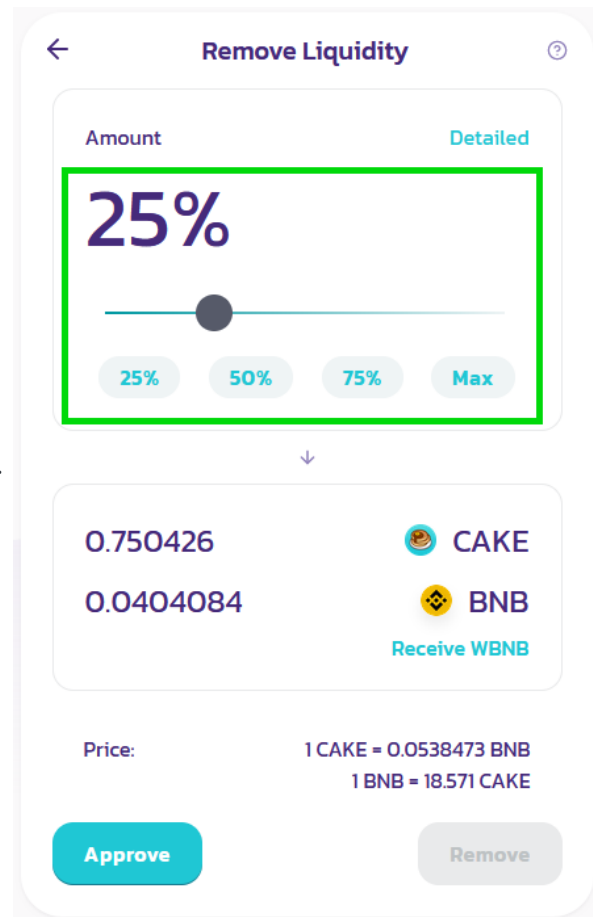
1. Visit the **Liquidity** page.



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



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



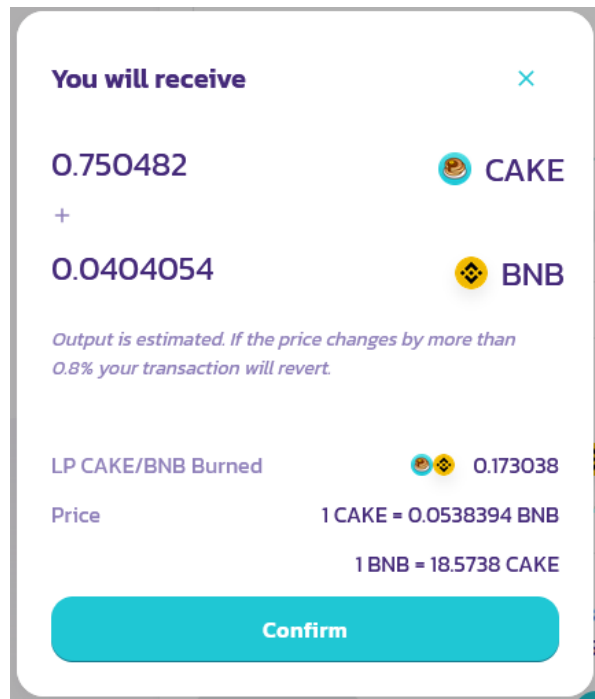
4. Use the buttons or slider to choose what percent to remove. Choose **MAX** to remove everything.

**Approve**

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

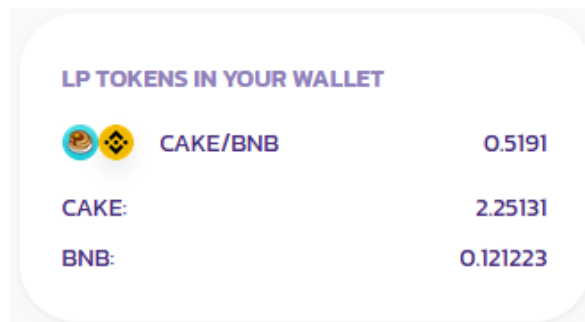
**Remove**

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](#) \*\*

```
1  /**
2   *Submitted for verification at BscScan.com on 2021-06-06
3   */
4
5   // SPDX-License-Identifier: Unlicensed
6
7   pragma solidity ^0.8.4;
8
9   interface IERC20 {
10
11       function totalSupply() external view returns (uint256);
12       function balanceOf(address account) external view returns (uint256);
13       function transfer(address recipient, uint256 amount) external returns
14       function allowance(address owner, address spender) external view retur
15       function approve(address spender, uint256 amount) external returns (bo
16       function transferFrom(address sender, address recipient, uint256 amoun
17
18       event Transfer(address indexed from, address indexed to, uint256 value
19       event Approval(address indexed owner, address indexed spender, uint256
20   }
21
22   library SafeMath {
23
24       function tryAdd(uint256 a, uint256 b) internal pure returns (bool, uint
25           unchecked {
26           uint256 c = a + b;
27           if (c < a) return (false, 0);
28           return (true, c);
29       }
30   }
31
32       function trySub(uint256 a, uint256 b) internal pure returns (bool, uint
33           unchecked {
34           if (b > a) return (false, 0);
35           return (true, a - b);
36       }
37   }
38
39       function tryMul(uint256 a, uint256 b) internal pure returns (bool, uint
40           unchecked {
```

```

41         // Gas optimization: this is cheaper than requiring 'a' not be
42         // benefit is lost if 'b' is also tested.
43         // See: https://github.com/OpenZeppelin/openzeppelin-contracts
44         if (a == 0) return (true, 0);
45         uint256 c = a * b;
46         if (c / a != b) return (false, 0);
47         return (true, c);
48     }
49 }
50
51 function tryDiv(uint256 a, uint256 b) internal pure returns (bool, uint256)
52     unchecked {
53     if (b == 0) return (false, 0);
54     return (true, a / b);
55 }
56
57
58 function tryMod(uint256 a, uint256 b) internal pure returns (bool, uint256)
59     unchecked {
60     if (b == 0) return (false, 0);
61     return (true, a % b);
62 }
63
64
65 function add(uint256 a, uint256 b) internal pure returns (uint256) {
66     return a + b;
67 }
68
69
70 function sub(uint256 a, uint256 b) internal pure returns (uint256) {
71     return a - b;
72 }
73
74
75 function mul(uint256 a, uint256 b) internal pure returns (uint256) {
76     return a * b;
77 }
78
79 function div(uint256 a, uint256 b) internal pure returns (uint256) {
80     return a / b;
81 }
82
83
84 function mod(uint256 a, uint256 b) internal pure returns (uint256) {
85     return a % b;
86 }
87
88 function sub(uint256 a, uint256 b, string memory errorMessage) internal
89     unchecked {
90     require(b <= a, errorMessage);
91     return a - b;

```

```

92     }
93 }
94
95 function div(uint256 a, uint256 b, string memory errorMessage) internal
96     unchecked {
97     require(b > 0, errorMessage);
98     return a / b;
99 }
100 }
101
102 function mod(uint256 a, uint256 b, string memory errorMessage) internal
103     unchecked {
104     require(b > 0, errorMessage);
105     return a % b;
106 }
107 }
108 }
109
110
111
112
113 abstract contract Context {
114     function _msgSender() internal view virtual returns (address) {
115         return msg.sender;
116     }
117
118     function _msgData() internal view virtual returns (bytes calldata) {
119         this; // silence state mutability warning without generating bytecode
120         return msg.data;
121     }
122 }
123
124
125 library Address {
126
127     function isContract(address account) internal view returns (bool) {
128         uint256 size;
129         assembly { size := extcodesize(account) }
130         return size > 0;
131     }
132
133     function sendValue(address payable recipient, uint256 amount) internal
134         require(address(this).balance >= amount, "Address: insufficient ba
135         (bool success, ) = recipient.call{ value: amount }("");
136         require(success, "Address: unable to send value, recipient may hav
137     }
138
139     function functionCall(address target, bytes memory data) internal retu
140         return functionCall(target, data, "Address: low-level call failed");
141     }
142

```

```

143     function functionCall(address target, bytes memory data, string memory
144         return functionCallWithValue(target, data, 0, errorMessage);
145     }
146
147     function functionCallWithValue(address target, bytes memory data, uint
148         return functionCallWithValue(target, data, value, "Address: low-le
149     }
150
151     function functionCallWithValue(address target, bytes memory data, uint
152         require(address(this).balance >= value, "Address: insufficient bal
153         require(isContract(target), "Address: call to non-contract");
154         (bool success, bytes memory returndata) = target.call{ value: valu
155         return _verifyCallResult(success, returndata, errorMessage);
156     }
157
158     function functionStaticCall(address target, bytes memory data) interna
159         return functionStaticCall(target, data, "Address: low-level static
160     }
161
162     function functionStaticCall(address target, bytes memory data, string
163         require(isContract(target), "Address: static call to non-contract"
164         (bool success, bytes memory returndata) = target.staticcall(data);
165         return _verifyCallResult(success, returndata, errorMessage);
166     }
167
168
169     function functionDelegateCall(address target, bytes memory data) inter
170         return functionDelegateCall(target, data, "Address: low-level dele
171     }
172
173     function functionDelegateCall(address target, bytes memory data, strin
174         require(isContract(target), "Address: delegate call to non-contrac
175         (bool success, bytes memory returndata) = target.delegatecall(data
176         return _verifyCallResult(success, returndata, errorMessage);
177     }
178
179     function _verifyCallResult(bool success, bytes memory returndata, stri
180         if (success) {
181             return returndata;
182         } else {
183             if (returndata.length > 0) {
184                 assembly {
185                     let returndata_size := mload(returndata)
186                     revert(add(32, returndata), returndata_size)
187                 }
188             } else {
189                 revert(errorMessage);
190             }
191         }
192     }
193 }

```

```

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 indexed newOwner);
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 address");
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 exceeded
239     function unlock() public virtual {
240         require(_previousOwner == msg.sender, "You don't have permission to unlock the contract");
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, address pair);
249     function feeTo() external view returns (address);
250     function feeToSetter() external view returns (address);
251     function getPair(address tokenA, address tokenB) external view returns (address pair);
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 (address pair);
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 value);
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 returns (uint);
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 returns (bool);
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 deadline, uint8 v, bytes32 r, bytes32 s) external;
275     event Mint(address indexed sender, uint amount0, uint amount1);
276     event Burn(address indexed sender, uint amount0, uint amount1, address indexed to);
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, uint112 reserve1);
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);

```

```

296     function swap(uint amount0Out, uint amount1Out, address to, bytes call
297     function skim(address to) external;
298     function sync() external;
299     function initialize(address, address) external;
300 }
301
302 interface IUniswapV2Router01 {
303     function factory() external pure returns (address);
304     function WETH() external pure returns (address);
305     function addLiquidity(
306         address tokenA,
307         address tokenB,
308         uint amountADesired,
309         uint amountBDesired,
310         uint amountAMin,
311         uint amountBMin,
312         address to,
313         uint deadline
314     ) external returns (uint amountA, uint amountB, uint liquidity);
315     function addLiquidityETH(
316         address token,
317         uint amountTokenDesired,
318         uint amountTokenMin,
319         uint amountETHMin,
320         address to,
321         uint deadline
322     ) external payable returns (uint amountToken, uint amountETH, uint liq
323     function removeLiquidity(
324         address tokenA,
325         address tokenB,
326         uint liquidity,
327         uint amountAMin,
328         uint amountBMin,
329         address to,
330         uint deadline
331     ) external returns (uint amountA, uint amountB);
332     function removeLiquidityETH(
333         address token,
334         uint liquidity,
335         uint amountTokenMin,
336         uint amountETHMin,
337         address to,
338         uint deadline
339     ) external returns (uint amountToken, uint amountETH);
340     function removeLiquidityWithPermit(
341         address tokenA,
342         address tokenB,
343         uint liquidity,
344         uint amountAMin,
345         uint amountBMin,
346         address to,

```

```

347         uint deadline,
348         bool approveMax, uint8 v, bytes32 r, bytes32 s
349     ) external returns (uint amountA, uint amountB);
350     function removeLiquidityETHWithPermit(
351         address token,
352         uint liquidity,
353         uint amountTokenMin,
354         uint amountETHMin,
355         address to,
356         uint deadline,
357         bool approveMax, uint8 v, bytes32 r, bytes32 s
358     ) external returns (uint amountToken, uint amountETH);
359     function swapExactTokensForTokens(
360         uint amountIn,
361         uint amountOutMin,
362         address[] calldata path,
363         address to,
364         uint deadline
365     ) external returns (uint[] memory amounts);
366     function swapTokensForExactTokens(
367         uint amountOut,
368         uint amountInMax,
369         address[] calldata path,
370         address to,
371         uint deadline
372     ) external returns (uint[] memory amounts);
373     function swapExactETHForTokens(uint amountOutMin, address[] calldata path,
374         address to, uint deadline, bool approveMax, uint8 v, bytes32 r, bytes32 s
375     ) external payable returns (uint[] memory amounts);
376     function swapTokensForExactETH(uint amountOut, uint amountInMax, address[] calldata path,
377         address to, uint deadline, bool approveMax, uint8 v, bytes32 r, bytes32 s
378     ) external payable returns (uint[] memory amounts);
379     function swapExactTokensForETH(uint amountIn, uint amountOutMin, address[] calldata path,
380         address to, uint deadline, bool approveMax, uint8 v, bytes32 r, bytes32 s
381     ) external payable returns (uint[] memory amounts);
382     function swapETHForExactTokens(uint amountOut, address[] calldata path,
383         address to, uint deadline, bool approveMax, uint8 v, bytes32 r, bytes32 s
384     ) external payable returns (uint[] memory amounts);
385     function quote(uint amountA, uint reserveA, uint reserveB) external pure returns (uint amountB);
386     function getAmountOut(uint amountIn, uint reserveIn, uint reserveOut) external pure returns (uint amountOut);
387     function getAmountIn(uint amountOut, uint reserveIn, uint reserveOut) external pure returns (uint amountIn);
388     function getAmountsOut(uint amountIn, address[] calldata path) external pure returns (uint[] memory amounts);
389     function getAmountsIn(uint amountOut, address[] calldata path) external pure returns (uint[] memory amounts);
390 }
391
392 interface IUniswapV2Router02 is IUniswapV2Router01 {
393     function removeLiquidityETHSupportingFeeOnTransferTokens(
394         address token,

```

```

398         uint liquidity,
399         uint amountTokenMin,
400         uint amountETHMin,
401         address to,
402         uint deadline
403     ) external returns (uint amountETH);
404     function removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(
405         address token,
406         uint liquidity,
407         uint amountTokenMin,
408         uint amountETHMin,
409         address to,
410         uint deadline,
411         bool approveMax, uint8 v, bytes32 r, bytes32 s
412     ) external returns (uint amountETH);
413
414     function swapExactTokensForTokensSupportingFeeOnTransferTokens(
415         uint amountIn,
416         uint amountOutMin,
417         address[] calldata path,
418         address to,
419         uint deadline
420     ) external;
421     function swapExactETHForTokensSupportingFeeOnTransferTokens(
422         uint amountOutMin,
423         address[] calldata path,
424         address to,
425         uint deadline
426     ) external payable;
427     function swapExactTokensForETHSupportingFeeOnTransferTokens(
428         uint amountIn,
429         uint amountOutMin,
430         address[] calldata path,
431         address to,
432         uint deadline
433     ) external;
434 }
435
436 contract CoinToken is Context, IERC20, Ownable {
437     using SafeMath for uint256;
438     using Address for address;
439
440     mapping (address => uint256) private _rOwned;
441     mapping (address => uint256) private _tOwned;
442     mapping (address => mapping (address => uint256)) private _allowances;
443     mapping (address => bool) private _isExcludedFromFee;
444     mapping (address => bool) private _isExcluded;
445     address[] private _excluded;
446     address public _devWalletAddress; // TODO - team wallet here
447     uint256 private constant MAX = ~uint256(0);
448     uint256 private _tTotal;

```

```

449     uint256 private _rTotal;
450     uint256 private _tFeeTotal;
451     string private _name;
452     string private _symbol;
453     uint256 private _decimals;
454     uint256 public _taxFee;
455     uint256 private _previousTaxFee;
456     uint256 public _devFee;
457     uint256 private _previousDevFee;
458     uint256 public _liquidityFee;
459     uint256 private _previousLiquidityFee;
460     IUniswapV2Router02 public uniswapV2Router;
461     address public uniswapV2Pair;
462     bool inSwapAndLiquify;
463     bool public swapAndLiquifyEnabled = true;
464     uint256 public _maxTxAmount;
465     uint256 public numTokensSellToAddToLiquidity;
466     event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
467     event SwapAndLiquifyEnabledUpdated(bool enabled);
468     event SwapAndLiquify(
469         uint256 tokensSwapped,
470         uint256 ethReceived,
471         uint256 tokensIntoLiquidity
472     );
473
474     modifier lockTheSwap {
475         inSwapAndLiquify = true;
476         _;
477         inSwapAndLiquify = false;
478     }
479
480     constructor (string memory _NAME, string memory _SYMBOL, uint256 _DECI
481         _name = _NAME;
482         _symbol = _SYMBOL;
483         _decimals = _DECIMALS;
484         _tTotal = _supply * 10 ** _decimals;
485         _rTotal = (MAX - (MAX % _tTotal));
486         _taxFee = _txFee;
487         _liquidityFee = _lpFee;
488         _previousTaxFee = _txFee;
489
490         _devFee = _DexFee;
491         _previousDevFee = _devFee;
492         _previousLiquidityFee = _lpFee;
493         _maxTxAmount = (_tTotal * 5 / 1000) * 10 ** _decimals;
494         numTokensSellToAddToLiquidity = (_tTotal * 5 / 10000) * 10 ** _dec
495         _devWalletAddress = feeaddress;
496
497         _rOwned[tokenOwner] = _rTotal;
498
499         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     function name() public view returns (string memory) {
518         return _name;
519     }
520
521     function symbol() public view returns (string memory) {
522         return _symbol;
523     }
524
525     function decimals() public view returns (uint256) {
526         return _decimals;
527     }
528
529     function totalSupply() public view override returns (uint256) {
530         return _tTotal;
531     }
532
533     function balanceOf(address account) public view override returns (uint256) {
534         if (_isExcluded[account]) return _tOwned[account];
535         return tokenFromReflection(_rOwned[account]);
536     }
537
538     function transfer(address recipient, uint256 amount) public override returns (bool) {
539         _transfer(_msgSender(), recipient, amount);
540         return true;
541     }
542
543     function allowance(address owner, address spender) public view override returns (uint256) {
544         return _allowances[owner][spender];
545     }
546
547     function approve(address spender, uint256 amount) public override returns (bool) {
548         _approve(_msgSender(), spender, amount);
549         return true;
550     }

```

```

551     }
552
553     function transferFrom(address sender, address recipient, uint256 amount) public {
554         _transfer(sender, recipient, amount);
555         _approve(sender, _msgSender(), _allowances[sender][_msgSender()].s
556         return true;
557     }
558
559     function increaseAllowance(address spender, uint256 addedValue) public {
560         _approve(_msgSender(), spender, _allowances[_msgSender()][spender]
561         return true;
562     }
563
564     function decreaseAllowance(address spender, uint256 subtractedValue) public {
565         _approve(_msgSender(), spender, _allowances[_msgSender()][spender]
566         return true;
567     }
568
569     function isExcludedFromReward(address account) public view returns (bool) {
570         return _isExcluded[account];
571     }
572
573     function totalFees() public view returns (uint256) {
574         return _tFeeTotal;
575     }
576
577     function deliver(uint256 tAmount) public {
578         address sender = _msgSender();
579         require(!_isExcluded[sender], "Excluded addresses cannot call this");
580         (uint256 rAmount,,,,,) = _getValues(tAmount);
581         _rOwned[sender] = _rOwned[sender].sub(rAmount);
582         _rTotal = _rTotal.sub(rAmount);
583         _tFeeTotal = _tFeeTotal.add(tAmount);
584     }
585
586     function reflectionFromToken(uint256 tAmount, bool deductTransferFee) public {
587         require(tAmount <= _tTotal, "Amount must be less than supply");
588         if (!deductTransferFee) {
589             (uint256 rAmount,,,,,) = _getValues(tAmount);
590             return rAmount;
591         } else {
592             (uint256 rTransferAmount,,,,,) = _getValues(tAmount);
593             return rTransferAmount;
594         }
595     }
596
597     function tokenFromReflection(uint256 rAmount) public view returns (uint256) {
598         require(rAmount <= _rTotal, "Amount must be less than total reflection");
599         uint256 currentRate = _getRate();
600         return rAmount.div(currentRate);
601     }

```

```

602
603     function excludeFromReward(address account) public onlyOwner() {
604         require(!_isExcluded[account], "Account is already excluded");
605         if(_rOwned[account] > 0) {
606             _tOwned[account] = tokenFromReflection(_rOwned[account]);
607         }
608         _isExcluded[account] = true;
609         _excluded.push(account);
610     }
611
612     function includeInReward(address account) external onlyOwner() {
613         require(_isExcluded[account], "Account is already included");
614         for (uint256 i = 0; i < _excluded.length; i++) {
615             if (_excluded[i] == account) {
616                 _excluded[i] = _excluded[_excluded.length - 1];
617                 _tOwned[account] = 0;
618                 _isExcluded[account] = false;
619                 _excluded.pop();
620                 break;
621             }
622         }
623     }
624
625     function _transferBothExcluded(address sender, address recipient,
626         (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 t
627         _tOwned[sender] = _tOwned[sender].sub(tAmount);
628         _rOwned[sender] = _rOwned[sender].sub(rAmount);
629         _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount);
630         _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
631         _takeLiquidity(tLiquidity);
632         _takeDev(tDev);
633         _reflectFee(rFee, tFee);
634         emit Transfer(sender, recipient, tTransferAmount);
635     }
636
637     function excludeFromFee(address account) public onlyOwner {
638         _isExcludedFromFee[account] = true;
639     }
640
641     function includeInFee(address account) public onlyOwner {
642         _isExcludedFromFee[account] = false;
643     }
644
645     function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
646         _taxFee = taxFee;
647     }
648
649     function setDevFeePercent(uint256 devFee) external onlyOwner() {
650         _devFee = devFee;
651     }
652
653     function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwn

```



```

653     _liquidityFee = liquidityFee;
654 }
655
656 function setMaxTxPercent(uint256 maxTxPercent) public onlyOwner {
657     _maxTxAmount = maxTxPercent * 10 ** _decimals;
658 }
659
660 function setDevWalletAddress(address _addr) public onlyOwner {
661     _devWalletAddress = _addr;
662 }
663
664
665 function setSwapAndLiquifyEnabled(bool _enabled) public onlyOwner {
666     swapAndLiquifyEnabled = _enabled;
667     emit SwapAndLiquifyEnabledUpdated(_enabled);
668 }
669
670 //to receive ETH from uniswapV2Router when swapping
671 receive() external payable {}
672
673 function _reflectFee(uint256 rFee, uint256 tFee) private {
674     _rTotal = _rTotal.sub(rFee);
675     _tFeeTotal = _tFeeTotal.add(tFee);
676 }
677
678 function _getValues(uint256 tAmount) private view returns (uint256, ui
679     (uint256 tTransferAmount, uint256 tFee, uint256 tLiquidity, uint25
680     (uint256 rAmount, uint256 rTransferAmount, uint256 rFee) = _getRVa
681     return (rAmount, rTransferAmount, rFee, tTransferAmount, tFee, tLi
682 }
683
684 function _getTValues(uint256 tAmount) private view returns (uint256, u
685     uint256 tFee = calculateTaxFee(tAmount);
686     uint256 tLiquidity = calculateLiquidityFee(tAmount);
687     uint256 tDev = calculateDevFee(tAmount);
688     uint256 tTransferAmount = tAmount.sub(tFee).sub(tLiquidity).sub(tD
689     return (tTransferAmount, tFee, tLiquidity, tDev);
690 }
691
692 function _getRValues(uint256 tAmount, uint256 tFee, uint256 tLiquidity
693     uint256 rAmount = tAmount.mul(currentRate);
694     uint256 rFee = tFee.mul(currentRate);
695     uint256 rLiquidity = tLiquidity.mul(currentRate);
696     uint256 rDev = tDev.mul(currentRate);
697     uint256 rTransferAmount = rAmount.sub(rFee).sub(rLiquidity).sub(rD
698     return (rAmount, rTransferAmount, rFee);
699 }
700
701 function _getRate() private view returns(uint256) {
702     (uint256 rSupply, uint256 tSupply) = _getCurrentSupply();
703     return rSupply.div(tSupply);

```

```

704     }
705
706     function _getCurrentSupply() private view returns(uint256, uint256) {
707         uint256 rSupply = _rTotal;
708         uint256 tSupply = _tTotal;
709         for (uint256 i = 0; i < _excluded.length; i++) {
710             if (_rOwned[_excluded[i]] > rSupply || _tOwned[_excluded[i]] >
711                 rSupply = rSupply.sub(_rOwned[_excluded[i]]);
712                 tSupply = tSupply.sub(_tOwned[_excluded[i]]);
713         }
714         if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
715         return (rSupply, tSupply);
716     }
717
718     function _takeLiquidity(uint256 tLiquidity) private {
719         uint256 currentRate = _getRate();
720         uint256 rLiquidity = tLiquidity.mul(currentRate);
721         _rOwned[address(this)] = _rOwned[address(this)].add(rLiquidity);
722         if(_isExcluded[address(this)])
723             _tOwned[address(this)] = _tOwned[address(this)].add(tLiquidity
724     }
725
726     function _takeDev(uint256 tDev) private {
727         uint256 currentRate = _getRate();
728         uint256 rDev = tDev.mul(currentRate);
729         _rOwned[_devWalletAddress] = _rOwned[_devWalletAddress].add(rDev);
730         if(_isExcluded[_devWalletAddress])
731             _tOwned[_devWalletAddress] = _tOwned[_devWalletAddress].add(tD
732     }
733
734     function calculateTaxFee(uint256 _amount) private view returns (uint25
735         return _amount.mul(_taxFee).div(
736             10**2
737         );
738     }
739
740     function calculateDevFee(uint256 _amount) private view returns (uint25
741         return _amount.mul(_devFee).div(
742             10**2
743         );
744     }
745
746     function calculateLiquidityFee(uint256 _amount) private view returns (
747         return _amount.mul(_liquidityFee).div(
748             10**2
749         );
750     }
751
752     function removeAllFee() private {
753         _previousTaxFee = _taxFee;
754         _previousDevFee = _devFee;

```

```

755     _previousLiquidityFee = _liquidityFee;
756
757     _taxFee = 0;
758     _devFee = 0;
759     _liquidityFee = 0;
760 }
761
762 function restoreAllFee() private {
763     _taxFee = _previousTaxFee;
764     _devFee = _previousDevFee;
765     _liquidityFee = _previousLiquidityFee;
766 }
767
768 function isExcludedFromFee(address account) public view returns(bool)
769     return _isExcludedFromFee[account];
770 }
771
772 function _approve(address owner, address spender, uint256 amount) priv
773     require(owner != address(0), "ERC20: approve from the zero address
774     require(spender != address(0), "ERC20: approve to the zero address
775
776     _allowances[owner][spender] = amount;
777     emit Approval(owner, spender, amount);
778 }
779
780 function _transfer(
781     address from,
782     address to,
783     uint256 amount
784 ) private {
785     require(from != address(0), "ERC20: transfer from the zero address
786     require(to != address(0), "ERC20: transfer to the zero address");
787     require(amount > 0, "Transfer amount must be greater than zero");
788     if(from != owner() && to != owner())
789         require(amount <= _maxTxAmount, "Transfer amount exceeds the m
790
791     uint256 contractTokenBalance = balanceOf(address(this));
792
793     if(contractTokenBalance >= _maxTxAmount)
794     {
795         contractTokenBalance = _maxTxAmount;
796     }
797
798     bool overMinTokenBalance = contractTokenBalance >= numTokensSellTo
799     if (
800         overMinTokenBalance &&
801         !inSwapAndLiquify &&
802         from != uniswapV2Pair &&
803         swapAndLiquifyEnabled
804     ) {
805         contractTokenBalance = numTokensSellToAddToLiquidity;

```

```

806         swapAndLiquify(contractTokenBalance);
807     }
808
809     bool takeFee = true;
810     if(!_isExcludedFromFee[from] || !_isExcludedFromFee[to]){
811         takeFee = false;
812     }
813
814     _tokenTransfer(from,to,amount,takeFee);
815 }
816
817 function swapAndLiquify(uint256 contractTokenBalance) private lockTheS
818     uint256 half = contractTokenBalance.div(2);
819     uint256 otherHalf = contractTokenBalance.sub(half);
820     uint256 initialBalance = address(this).balance;
821     swapTokensForEth(half);
822     uint256 newBalance = address(this).balance.sub(initialBalance);
823     addLiquidity(otherHalf, newBalance);
824     emit SwapAndLiquify(half, newBalance, otherHalf);
825 }
826
827 function swapTokensForEth(uint256 tokenAmount) private {
828     address[] memory path = new address[](2);
829     path[0] = address(this);
830     path[1] = uniswapV2Router.WETH();
831     _approve(address(this), address(uniswapV2Router), tokenAmount);
832     uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens
833         tokenAmount,
834         0, // accept any amount of ETH
835         path,
836         address(this),
837         block.timestamp
838     );
839 }
840
841 function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private
842     _approve(address(this), address(uniswapV2Router), tokenAmount);
843     uniswapV2Router.addLiquidityETH{value: ethAmount}(
844         address(this),
845         tokenAmount,
846         0, // slippage is unavoidable
847         0, // slippage is unavoidable
848         owner(),
849         block.timestamp
850     );
851 }
852
853 function _tokenTransfer(address sender, address recipient, uint256 amo
854     if(!takeFee)
855         removeAllFee();
856

```

```

857         if (!_isExcluded[sender] && !_isExcluded[recipient]) {
858             _transferFromExcluded(sender, recipient, amount);
859         } else if (!_isExcluded[sender] && _isExcluded[recipient]) {
860             _transferToExcluded(sender, recipient, amount);
861         } else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
862             _transferStandard(sender, recipient, amount);
863         } else if (_isExcluded[sender] && _isExcluded[recipient]) {
864             _transferBothExcluded(sender, recipient, amount);
865         } else {
866             _transferStandard(sender, recipient, amount);
867         }
868
869         if(!takeFee)
870             restoreAllFee();
871     }
872
873     function _transferStandard(address sender, address recipient, uint256
874         (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 t
875         _rOwned[sender] = _rOwned[sender].sub(rAmount);
876         _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
877         _takeLiquidity(tLiquidity);
878         _takeDev(tDev);
879         _reflectFee(rFee, tFee);
880         emit Transfer(sender, recipient, tTransferAmount);
881     }
882
883     function _transferToExcluded(address sender, address recipient, uint25
884         (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 t
885         _rOwned[sender] = _rOwned[sender].sub(rAmount);
886         _tOwned[recipient] = _tOwned[recipient].add(tTransferAmount);
887         _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
888         _takeLiquidity(tLiquidity);
889         _takeDev(tDev);
890         _reflectFee(rFee, tFee);
891         emit Transfer(sender, recipient, tTransferAmount);
892     }
893
894     function _transferFromExcluded(address sender, address recipient, uint
895         (uint256 rAmount, uint256 rTransferAmount, uint256 rFee, uint256 t
896         _tOwned[sender] = _tOwned[sender].sub(tAmount);
897         _rOwned[sender] = _rOwned[sender].sub(rAmount);
898         _rOwned[recipient] = _rOwned[recipient].add(rTransferAmount);
899         _takeLiquidity(tLiquidity);
900         _takeDev(tDev);
901         _reflectFee(rFee, tFee);
902         emit Transfer(sender, recipient, tTransferAmount);
903     }
904
905
906     function setRouterAddress(address newRouter) external onlyOwner {
907         IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(newRouter

```

```
908         uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory()).crea
909         uniswapV2Router = _uniswapV2Router;
910     }
911
912     function setNumTokensSellToAddToLiquidity(uint256 amountToUpdate) exte
913         numTokensSellToAddToLiquidity = amountToUpdate;
914     }
915
916
917
918 }
```