Reward Sharing

Nhan Cao

--

nhancv92@gmail.com

***Keywords*—Dividend, Reward distribution, Reflection, Smart contract**

I. CONTEXT

BUY FROM DEX						
1	BOB	Buy 103 > Tax 3 > Pool = 3, balance = 100, total staked = 100				
2	ADAM	Buy 206 > Tax 6 > Pool = 9, balance = 200, total staked = 300				
3	CAROL	Buy 309 > Tax 9 > Pool = 18, balance = 300, total staked = 600				
4	ERIC	Buy 103 > Tax 3 > Pool = 21, balance = 100, total staked = 700				
FORMULA		Total buy: 103 + 206 + 309 + 103 = 700 + 21 = 721				
1	BOB	100 * 3/100	100 * 6/300	100 * 9/600	100 * 3/700	3.0 + 2.0 + 1.5 + 0.43 = 6.93
2	ADAM	0	200 * 6/300	200 * 9/600	200 * 3/700	0.0 + 4.0 + 3.0 + 0.86 = 7.86
3	CAROL	0	0	300 * 9/600	300 * 3/700	0.0 + 0.0 + 4.5 + 1.28 = 5.78
4	ERIC	0	0	0	100 * 3/700	0.0 + 0.0 + 0.0 + 0.43 = 0.43
					Total = 6.93 + 7.86 + 5.78 + 0.43 = 21 = pool size	
TRANSFER						
1	BOB	Buy 103 > Tax 3 > Pool = 3, balance = 100, total staked = 100				
2	ADAM	Buy 206 > Tax 6 > Pool = 9, balance = 200, total staked = 300				
3	BOB send 50 coins to ADAM => CONVERT TO BUY CASE: ADAM BUY 50 from BOB					
3.1	BOB	Balance = 100 – 50 = 50				
3.2	ADAM	Same BUY 50 from DEX > Tax = 1.46 > Pool = 10.46, balance = 200 + 48.54 = 248.54, total staked = 298.54				
FORMULA						
1	BOB	100 * 3/100	100 * 6/300	50 * 1.46/298.54	3.0 + 2.0 + 0.24 = 5.24	
2	ADAM	0	200 * 6/300	248.54 * 1.46/298.54	0.0 + 4.0 + 1.22 = 5.22	
					Total = 5.24 + 5.22 = 10.46 = pool size	
SELL TO DEX						
1	BOB	Buy 103 > Tax 3 > Pool = 3, balance = 100, total staked = 100				
2	ADAM	Buy 206 > Tax 6 > Pool = 9, balance = 200, total staked = 300				
3	BOB SELL 50 coins => CONVERT TO TRANSFER CASE: BOB TRANSFER to LP					
3.1	BOB	Sell 50 > Tax 1.5 > Pool 10.5, balance = 50, total staked = 250				
FORMULA						
1	BOB	100 * 3/100	100 * 6/300	50 * 1.5/250	3.0 + 2.0 + 0.3 = 5.3	
2	ADAM	0	200 * 6/300	200 * 1.5/250	0.0 + 4.0 + 1.2 = 5.2	
					Total = 5.3 + 5.2 = 10.5 = pool size	

II. ALGORITHM

Global	// Global Variables
	l: uint = 0 // Portion map length
	r: map (uint -> uint) // Portion reward mapping
	urc: map (address -> uint) // User's reward accumulation
	up: map (address -> uint) // User's portion reward offset
	ud: map (address -> uint) // User's debt
	t: uint // Total staked
	pool: uint // Total pool reward
	distributed: uint // Distributed reward
	fee: uint // fee percent

Transfer (from_address, to_address, amount)	**# -------**
	# Token transfer function: transfer(from_address, to_address, amount)
	// Update portion anytime coin changes: buy, sell, transfer
	tp: address // tax payer
	
	tax = fee * amount;
	after_tax = amount - tax;
	pool = pool + tax;
	
	if (is_buy) {
	  t = t + after_tax;
	  tp = to_address;
	
	} else if (is_transfer) {
	  t = t - tax;
	  tp = to_address;
	
	  urc[from_address] = urc[from_address] +
	get_portion_reward(from_address);
	  up[from_address] = l + 1;
	
	} else {
	  t = t - amount;
	  tp = from_address;
	
	}
	
	// Update user offset and portion first
	urc[tp] = urc[tp] + get_portion_reward(tp);
	up[tp] = l + 1;
	
	// Update reward portion
	r[l+1] = r[l] + tax / t;
	l = l + 1;
	
	// Call native transfer
	super.transfer(from_address, address(this), tax);
	super.transfer(from_address, to_address, after_tax);

get_portion_reward (user_address)	**# -------** **# Get user's portion reward amount function: get_portion_reward(user_address)**

	// Get pending reward of user address
	portion = up[user_address];
	if (portion == 0) portion = 1;
	return (r[l] - r[portion - 1]) * balanceOf(user_address);

get_max_reward (user_address)	**# -------**
	# Get user's max reward amount function:
	get_max_reward(user_address)
	// Get max reward of user address
	return get_portion_reward(user_address) + urc[user_address];

get_pending_reward (user_address)	**# -------**
	# Get user's pending reward amount function:
	get_pending_reward(user_address)
	// Get pending reward of user address
	total = get_max_reward(user_address);
	if (total > ud[user_address]) return total - ud[user_address];
	return 0;

withdraw_pending_reward (user_address)	**# -------**
	# Withdraw user's pending reward amount function:
	withdraw_pending_reward(user_address)
	// Withdraw reward
	reward = get_pending_reward(user_address);
	if (reward > 0) {
	  distributed = distributed + reward;
	  ud[user_address] = ud[user_address] + reward;
	
	  … do something with reward …
	}

III. JAVASCRIPT

```
export class AppService {
  // Native transfer
  balance: { [index: string]: number } = {};

  _transfer(from: string, to: string, amount: number): void {
    this.balance[from] -= amount;
    this.balance[to] += amount;
  }

  l: number = 0;
  r: { [index: number]: number } = {};
  urc: { [index: string]: number } = {};
  up: { [index: string]: number } = {};
  ud: { [index: string]: number } = {};
  t: number = 0;
  pool: number = 0;
  distributed: number = 0;
  fee: number = 3 / 103; // 2.9%

  // reset state
  reset(): void {
    this.r = {0: 0};
    this.l = 0;
```

```
    this.urc = {};
    this.up = {};
    this.ud = {};
    this.t = 0;
    this.pool = 0;
    this.balance = {'bob': 0, 'adam': 0, 'carol': 0, 'eric': 0};
  }

  // Update reward in transfer action
  token_transfer(type: string, from_address: string, to_address: string, amount: number):
void {
    console.log(`type: ${type}, from: ${from_address}, to: ${to_address}`);
    let tp: string;

    const tax: number = this.fee * amount;
    const after_tax: number = amount - tax;
    this.pool = this.pool + tax;

    if (type === 'buy') {
      this.t = this.t + after_tax;
      tp = to_address;

    } else if (type === 'transfer') {
      this.t = this.t - tax;
      tp = to_address;

      this.urc[from_address] = (this.urc[from_address] ?? 0)  +
this.get_portion_reward(from_address);
      this.up[from_address] = this.l + 1;

    } else {
      this.t = this.t - amount;
      tp = from_address;

    }

    // Update user offset and portion first
    this.urc[tp] = (this.urc[tp] ?? 0) + this.get_portion_reward(tp);
    this.up[tp] = this.l + 1;

    // Update reward portion
    this.r[this.l + 1] = this.r[this.l] + tax/this.t;
    this.l = this.l + 1;

    // Call super native transfer
    this._transfer(from_address, 'pool', tax);
    this._transfer(from_address, to_address, after_tax);
  }

  // Get user portion reward function
  get_portion_reward(user_address: string): number {
    let portion = (this.up[user_address] ?? 0);
    if(portion == 0) {
      portion = 1;
    }
    return (this.r[this.l] - this.r[portion - 1]) * this.balance[user_address];
  }

  // Get user max reward function
  get_max_reward(user_address: string): number {
    return this.get_portion_reward(user_address) + (this.urc[user_address] ?? 0);
  }

  // Get user's pending reward amount function
  get_pending_reward(user_address: string): number {
```

```
    const total = this.get_max_reward(user_address);
    if(total > (this.ud[user_address] ?? 0)) {
      return total - (this.ud[user_address] ?? 0);
    }
    return 0;
  }

  // Withdraw user's pending reward amount function
  withdraw_pending_reward(user_address: string): void {
    const reward = this.get_pending_reward(user_address);
    if(reward > 0) {
      this.distributed += reward;
      this.ud[user_address] = (this.ud[user_address] ?? 0) + reward;
      console.log(`=> ${user_address} withdraw: ${reward}`);
    } else {
      console.log(`=> ${user_address} withdraw: empty`);
    }
  }
}
```

IV. Output

```
const app = await NestFactory.createApplicationContext(AppModule);
const appService = app.get(AppService);

const printRewards = (isFull: boolean = false): void => {
  console.log(`Bob reward: ${appService.get_pending_reward('bob')}, balance:
${appService.balance['bob']}`);
  console.log(`Adam reward: ${appService.get_pending_reward('adam')}, balance:
${appService.balance['adam']}`);
  if(isFull) {
    console.log(`Carol reward: ${appService.get_pending_reward('carol')}, balance:
${appService.balance['carol']}`);
    console.log(`Eric reward: ${appService.get_pending_reward('eric')}, balance:
${appService.balance['eric']}`);
  }
}

const printPoolSizeAndStaked = (isDistributed: boolean = false): void => {
  console.log('Pool size:', appService.pool);
  console.log('Total staked:', appService.t);
  if(isDistributed) {
    console.log('Distributed size:', appService.distributed);
  }
}

// -----
appService.reset();
// Buy
console.log('---BUY---');
appService.token_transfer('buy', 'dex', 'bob', 103);
appService.token_transfer('buy', 'dex', 'adam', 206);
appService.token_transfer('buy', 'dex', 'carol', 309);
appService.token_transfer('buy', 'dex', 'eric', 103);
// Log
printPoolSizeAndStaked();
printRewards(true);

// -----
appService.reset();
// Transfer
console.log('---TRANSFER---');
```

```
appService.token_transfer('buy', 'dex', 'bob', 103);
appService.token_transfer('buy', 'dex', 'adam', 206);
appService.token_transfer('transfer', 'bob', 'adam', 50);
// Log
printPoolSizeAndStaked();
printRewards();

// -----
appService.reset();
// Sell
console.log('---SELL---');
appService.token_transfer('buy', 'dex', 'bob', 103);
appService.token_transfer('buy', 'dex', 'adam', 206);
appService.token_transfer('sell', 'bob', 'dex', 50);
// Log
printPoolSizeAndStaked();
printRewards();

// -----
appService.reset();
// Withdraw
console.log('---WITHDRAW---');
appService.token_transfer('buy', 'dex', 'bob', 103);
printRewards();
appService.withdraw_pending_reward('bob'); // = 3

appService.token_transfer('buy', 'dex', 'adam', 206);
printRewards();
appService.withdraw_pending_reward('bob'); // = 3 + 2 - 3 = 2
appService.withdraw_pending_reward('adam'); // = 4

appService.token_transfer('transfer', 'bob', 'adam', 50);
printRewards();
appService.withdraw_pending_reward('bob'); // = 3 + 2 + 0.24 - 5 = 0.24
appService.withdraw_pending_reward('adam'); // = 4 + 1.22 - 4 = 1.22
printPoolSizeAndStaked();

appService.token_transfer('sell', 'bob', 'dex', 50);
printRewards();
appService.withdraw_pending_reward('bob'); // = 0 (balance = 0)
appService.withdraw_pending_reward('adam'); // = 4 + 1.22 + 1.46 - 5.22 = 1.46
printPoolSizeAndStaked();

appService.token_transfer('buy', 'dex', 'adam', 206);
printRewards();
appService.withdraw_pending_reward('bob'); // = 0 (balance = 0)
appService.withdraw_pending_reward('adam'); // = 4 + 1.22 + 1.46 + 6 - 6.68 = 6
printPoolSizeAndStaked(true);

==========

---BUY---
type: buy, from: dex, to: bob
type: buy, from: dex, to: adam
type: buy, from: dex, to: carol
type: buy, from: dex, to: eric
Pool size: 21
Total staked: 700
Bob reward: 6.928571428571429, balance: 100
Adam reward: 7.857142857142857, balance: 200
Carol reward: 5.785714285714285, balance: 300
Eric reward: 0.42857142857142816, balance: 100
```

```
---TRANSFER---
type: buy, from: dex, to: bob
type: buy, from: dex, to: adam
type: transfer, from: bob, to: adam
Pool size: 10.45631067961165
Total staked: 298.54368932038835
Bob reward: 5.2439024390243905, balance: 50
Adam reward: 5.21240824058726, balance: 248.54368932038835
---SELL---
type: buy, from: dex, to: bob
type: buy, from: dex, to: adam
type: sell, from: bob, to: dex
Pool size: 10.45631067961165
Total staked: 250
Bob reward: 5.29126213592233, balance: 50
Adam reward: 5.165048543689322, balance: 200
---WITHDRAW---
type: buy, from: dex, to: bob
Bob reward: 3, balance: 100
Adam reward: 0, balance: 0
=> bob withdraw: 3
type: buy, from: dex, to: adam
Bob reward: 2, balance: 100
Adam reward: 4.000000000000001, balance: 200
=> bob withdraw: 2
=> adam withdraw: 4.000000000000001
type: transfer, from: bob, to: adam
Bob reward: 0.24390243902439046, balance: 50
Adam reward: 1.2124082405872594, balance: 248.54368932038835
=> bob withdraw: 0.24390243902439046
=> adam withdraw: 1.2124082405872594
Pool size: 10.45631067961165
Total staked: 298.54368932038835
type: sell, from: bob, to: dex
Bob reward: 0, balance: 0
Adam reward: 1.4563106796116507, balance: 248.54368932038835
=> bob withdraw: empty
=> adam withdraw: 1.4563106796116507
Pool size: 11.912621359223301
Total staked: 248.54368932038835
type: buy, from: dex, to: adam
Bob reward: 0, balance: 0
Adam reward: 6.000000000000002, balance: 448.54368932038835
=> bob withdraw: empty
=> adam withdraw: 6.000000000000002
Pool size: 17.9126213592233
Total staked: 448.54368932038835
Distributed size: 17.9126213592233
```

V. SOLIDITY VERSION

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;
```

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

import "@openzeppelin/contracts/access/Ownable.sol";

// @author nhancv
// @title Reward sharing implementation -
// see https://github.com/nhancv/nideas/blob/main/RewardSharing.pdf
contract RewardSharing is Ownable {
  uint internal constant RATE_NOMINATOR = 10000;
  uint internal constant PRECISION_FACTOR = 1e36;

  // Pool structure
  struct PoolInfo {
    uint l; // Portion map length
    mapping(uint => uint) r; // Portion reward mapping
    mapping(address => uint) urc; // User's reward accumulation
    mapping(address => uint) up; // User's portion reward offset
    mapping(address => uint) ud; // User's debt
    mapping(address => uint) ub; // User's balance
    uint t; // Total staked
    uint pool; // Total pool reward
    uint distributed; // Distributed reward
    uint fee; // fee percent, ex: 100 mean 1%
  }

  // Pool array
  PoolInfo[] public pools;

  /**
  * @dev get the pools Length
  */
  function getPoolsLength() public view returns (uint) {
    return pools.length;
  }

  /**
  * @dev remove pool
  * @param fee fee of pool
  */
  function addPool(uint fee) onlyOwner public {
    uint256 idx = getPoolsLength();
    pools.push();

    PoolInfo storage info = pools[idx];
    info.fee = fee;
  }

  /**
  * @dev update pool
  * @param index index of pool
  * @param fee pool fee
  */
  function updatePool(uint index, uint fee) onlyOwner public {
    PoolInfo storage info = pools[index];
    info.fee = fee;
  }

  /**
  * @dev remove pool
  * @param index index of pool
  */
  function removePool(uint index) onlyOwner public {
    delete pools[index];
  }
```

```
  /**
  * @dev update portion reward. Call this function before real transfer token
  * @param index pool index
  * @param fromAddress from address
  * @param toAddress to address
  * @param amount amount of tokens
  * @param code transfer type: 0: transfer, 1: buy, 2: sell
  * @param airdrop in case owner send to user, not apply tax but save total staked
  */
  function _updatePortion(uint index, address fromAddress, address toAddress, uint amount,
uint code, bool airdrop) internal returns (uint _tax, uint _afterTax) {
    PoolInfo storage info = pools[index];

    address tp;
    uint tax = 0;
    if(!airdrop) tax = (info.fee * amount) / RATE_NOMINATOR;
    uint afterTax = amount - tax;
    info.pool = info.pool + tax;

    if (code == 1) {
      info.t = info.t + afterTax;
      tp = toAddress;

    } else if (code == 0) {
      if (info.t >= tax) info.t = info.t - tax;
      tp = toAddress;

      info.urc[fromAddress] = info.urc[fromAddress] + getPortionReward(index,
fromAddress);
      info.up[fromAddress] = info.l + 1;

    } else {
      if (info.t >= amount) info.t = info.t - amount;
      tp = fromAddress;

    }

    // Update user offset and portion first
    info.urc[tp] = info.urc[tp] + getPortionReward(index, tp);
    info.up[tp] = info.l + 1;

    // Update reward portion
    uint _t = 0;
    if (info.t > 0) _t = tax * PRECISION_FACTOR / info.t;
    info.r[info.l + 1] = info.r[info.l] + _t;
    info.l = info.l + 1;

    // Update user balance
    _updateBalance(index, fromAddress, address(this), tax);
    _updateBalance(index, fromAddress, toAddress, afterTax);

    return (tax, afterTax);
  }

  /**
  * @dev Update user balance
  */
  function _updateBalance(uint index, address fromAddress, address toAddress, uint amount)
internal virtual {
    PoolInfo storage info = pools[index];
    if (info.ub[fromAddress] >= amount) {
      info.ub[fromAddress] -= amount;
    } else {
      info.ub[fromAddress] = 0;
```

```
    }

    info.ub[toAddress] += amount;
  }

  /**
  * @dev Get user balance
  */
  function _userBalance(uint index, address userAddress) internal view returns (uint
balance) {
    PoolInfo storage info = pools[index];
    return info.ub[userAddress];
  }

  /**
  * @dev Get user portion reward function
  */
  function getPortionReward(uint index, address userAddress) public view returns (uint
portionReward) {
    PoolInfo storage info = pools[index];
    uint portion = info.up[userAddress];
    if (portion == 0) {
      portion = 1;
    }
    if (info.r[info.l] >= info.r[portion - 1]) {
      return (info.r[info.l] - info.r[portion - 1]) * _userBalance(index, userAddress);
    }
    return 0;
  }

  /**
  * @dev Get user max reward function
  */
  function getMaxReward(uint index, address userAddress) public view returns (uint
maxReward) {
    PoolInfo storage info = pools[index];
    return (getPortionReward(index, userAddress) + info.urc[userAddress]) /
PRECISION_FACTOR;
  }

  /**
  * @dev Get user's pending reward amount function
  */
  function getPendingReward(uint index, address userAddress) public view returns (uint
pendingReward) {
    PoolInfo storage info = pools[index];
    uint total = getMaxReward(index, userAddress);
    if (total > info.ud[userAddress]) {
      return total - info.ud[userAddress];
    }
    return 0;
  }

  /**
  * @dev Withdraw user's pending reward amount function.
  * This function get pending reward and update user debt after withdraw
  * Need implement new withdraw function in main contract.
  * Call this function to get reward and send token as your logic
  */
  function _withdrawPendingReward(uint index, address userAddress) internal returns (uint
withdrawnReward) {
    uint reward = getPendingReward(index, userAddress);
    PoolInfo storage info = pools[index];
    info.distributed += reward;
    info.ud[userAddress] += reward;
```

```
    return reward;
  }

}


=======
Compiling your contracts...
===========================
> Compiling ./src/active/TokenPresenter.sol
> Compiling ./src/active/utils/RewardSharing.sol
> Artifacts written to /var/folders/pv/_n9wn10539z4gq_17z945k300000gn/T/test--10423-
4bK2A5isqI3Y
> Compiled successfully using:
   - solc: 0.8.4+commit.c7e474f2.Emscripten.clang



  Contract: reflection.test
Owner: 0xF7e008130254b4bD272166dafeF52C65915D18e8
[ADD_LP] owner is adding liquidity of 100000 Reward1 and 100 BNB
BigNumber.toString does not accept any parameters; base-10 is assumed
[ADD_LP] owner is adding liquidity of 100000 N07 and 100 BNB
    initialize
      ✓ Verify init config (3769ms)
    logic
Owner: 0xF7e008130254b4bD272166dafeF52C65915D18e8
[ADD_LP] owner is adding liquidity of 100000 Reward1 and 100 BNB
[ADD_LP] owner is adding liquidity of 100000 N07 and 100 BNB
[BUY_EXACT_TOKENS] bob is buying worth of 103 Tokens
Estimated input: 0.103312825035437804
pool size: 3.0076
pool staked: 99.9924
pool distributed: 0.0
[bob]pendingReward: 3.007599999999999999
[bob]tokenBalance: 99.9924
[bob]rewardBalance: 0.0
bob CLAIM
pool size: 3.0076
pool staked: 99.9924
pool distributed: 3.007599999999999999
[bob]pendingReward: 0.0
[bob]tokenBalance: 99.9924
[bob]rewardBalance: 3.001587982910370198
[BUY_EXACT_TOKENS] adam is buying worth of 206 Tokens
Estimated input: 0.207254049949181209
[bob]pendingReward: 2.005066666666666667
[bob]tokenBalance: 99.9924
[bob]rewardBalance: 3.001587982910370198
[adam]pendingReward: 4.010133333333333333
[adam]tokenBalance: 199.9848
[adam]rewardBalance: 0.0
bob CLAIM
[bob]pendingReward: 0.0
[bob]tokenBalance: 99.9924
[bob]rewardBalance: 5.01073152988555529
[adam]pendingReward: 4.010133333333333333
[adam]tokenBalance: 199.9848
```

```
[adam]rewardBalance: 0.0
pool size: 9.0228
pool staked: 299.9772
pool distributed: 5.012666666666666666
[BUY_EXACT_TOKENS] carol is buying worth of 309 Tokens
Estimated input: 0.312480734533329329
[BUY_EXACT_TOKENS] eric is buying worth of 103 Tokens
Estimated input: 0.104593129452477457
[bob]pendingReward: 1.933457142857142857
[bob]tokenBalance: 99.9924
[bob]rewardBalance: 5.01073152988555529
[adam]pendingReward: 7.877047619047619047
[adam]tokenBalance: 199.9848
[adam]rewardBalance: 0.0
[carol]pendingReward: 5.800371428571428571
[carol]tokenBalance: 299.9772
[carol]rewardBalance: 0.0
[eric]pendingReward: 0.429657142857142857
[eric]tokenBalance: 99.9924
[eric]rewardBalance: 0.0
pool size: 21.0532
pool staked: 699.9468
pool distributed: 5.012666666666666666
bob CLAIM
adam CLAIM
carol CLAIM
eric CLAIM
[bob]pendingReward: 0.0
[bob]tokenBalance: 99.9924
[bob]rewardBalance: 6.964093968644591065
[adam]pendingReward: 0.0
[adam]tokenBalance: 199.9848
[adam]rewardBalance: 7.956569325178487977
[carol]pendingReward: 0.0
[carol]tokenBalance: 299.9772
[carol]rewardBalance: 5.857311779031531119
[eric]pendingReward: 0.0
[eric]tokenBalance: 99.9924
[eric]rewardBalance: 0.433820380346272228
pool size: 21.0532
pool staked: 699.9468
pool distributed: 21.053199999999999998
      ✓ buy (11658ms)
Owner: 0xF7e008130254b4bD272166dafeF52C65915D18e8
[ADD_LP] owner is adding liquidity of 100000 Reward1 and 100 BNB
[ADD_LP] owner is adding liquidity of 100000 N07 and 100 BNB
[BUY_EXACT_TOKENS] bob is buying worth of 103 Tokens
Estimated input: 0.103312825035437804
[BUY_EXACT_TOKENS] adam is buying worth of 206 Tokens
Estimated input: 0.207266530150884673
[TRANSFER_TOKEN] From bob to adam for 50 tokens
[bob]pendingReward: 5.257171519318373168
[bob]tokenBalance: 49.9924
[bob]rewardBalance: 0.0
[adam]pendingReward: 5.225628480681626831
[adam]tokenBalance: 248.5248
[adam]rewardBalance: 0.0
pool size: 10.4828
```

```
pool staked: 298.5172
pool distributed: 0.0
      ✓ transfer (5505ms)
Owner: 0xF7e008130254b4bD272166dafeF52C65915D18e8
[ADD_LP] owner is adding liquidity of 100000 Reward1 and 100 BNB
[ADD_LP] owner is adding liquidity of 100000 N07 and 100 BNB
[BUY_EXACT_TOKENS] bob is buying worth of 103 Tokens
Estimated input: 0.103312825035437804
[BUY_EXACT_TOKENS] adam is buying worth of 206 Tokens
Estimated input: 0.207266530150884673
[SELL] bob is selling 50 tokens
BNB snapshot: 999997.277441221110303483
Estimated output: 0.050185008455145256
[bob]pendingReward: 5.304648911447390668
[bob]tokenBalance: 49.9924
[bob]rewardBalance: 0.0
[adam]pendingReward: 5.178151088552609331
[adam]tokenBalance: 199.9848
[adam]rewardBalance: 0.0
pool size: 10.4828
pool staked: 249.9772
pool distributed: 0.0
      ✓ sell (5705ms)
Owner: 0xF7e008130254b4bD272166dafeF52C65915D18e8
[ADD_LP] owner is adding liquidity of 100000 Reward1 and 100 BNB
[ADD_LP] owner is adding liquidity of 100000 N07 and 100 BNB
[BUY_EXACT_TOKENS] bob is buying worth of 103 Tokens
Estimated input: 0.103312825035437804
[BUY_EXACT_TOKENS] adam is buying worth of 206 Tokens
Estimated input: 0.207266530150884673
[TRANSFER_TOKEN] From bob to adam for 50 tokens
[SELL] bob is selling 49.9924 tokens
BNB snapshot: 999997.205517334021242901
Estimated output: 0.050177384149600891
[BUY_EXACT_TOKENS] adam is buying worth of 206 Tokens
Estimated input: 0.207923245680291133
[bob]pendingReward: 5.257171519318373168
[bob]tokenBalance: 0.0
[bob]rewardBalance: 0.0
[adam]pendingReward: 12.700606560681626831
[adam]tokenBalance: 448.5096
[adam]rewardBalance: 0.0
pool size: 17.95777808
pool staked: 448.5096
pool distributed: 0.0
bob CLAIM
adam CLAIM
[bob]pendingReward: 0.0
[bob]tokenBalance: 0.0
[bob]rewardBalance: 5.284859663537046342
[adam]pendingReward: 0.0
[adam]tokenBalance: 448.5096
[adam]rewardBalance: 12.762892311784533734
pool size: 17.95777808
pool staked: 448.5096
pool distributed: 17.957778079999999999
      ✓ combine (7705ms)
```

```
Owner: 0xF7e008130254b4bD272166dafeF52C65915D18e8
[ADD_LP] owner is adding liquidity of 100000 Reward1 and 100 BNB
[ADD_LP] owner is adding liquidity of 100000 N07 and 100 BNB
[ADD_LP] owner is adding liquidity of 100000 Reward2 and 100 BNB
[BUY_EXACT_TOKENS] bob is buying worth of 103 Tokens
Estimated input: 0.103312825035437804
[BUY_EXACT_TOKENS] adam is buying worth of 206 Tokens
Estimated input: 0.207266530150884673
[TRANSFER_TOKEN] From bob to adam for 50 tokens
[SELL] bob is selling 43.9772 tokens
BNB snapshot: 999997.121232725040114541
Estimated output: 0.044142583271503352
[BUY_EXACT_TOKENS] adam is buying worth of 206 Tokens
Estimated input: 0.207958306302054554
bob CLAIM
adam CLAIM
POOL: 0
[bob]pendingReward: 0.0
[bob]tokenBalance: 0.0
[bob]rewardBalance: 5.396289108422247945
[adam]pendingReward: 0.0
[adam]tokenBalance: 421.5288
[adam]rewardBalance: 12.475359590232993053
pool size: 17.78213424
pool staked: 454.5248
pool distributed: 17.782134239999999999
POOL: 1
[bob]pendingReward: 0.0
[bob]tokenBalance: 0.0
[bob]rewardBalance: 10.671317769274254885
[adam]pendingReward: 0.0
[adam]tokenBalance: 421.5288
[adam]rewardBalance: 25.054443800364667268
pool size: 35.56426848
pool staked: 438.0268
pool distributed: 35.564268479999999999
      ✓ 2 pools (16454ms)


  6 passing (2m)
```