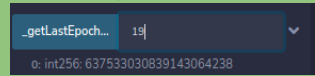
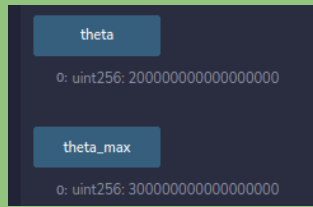
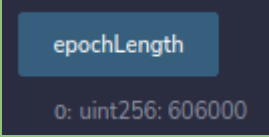
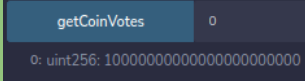
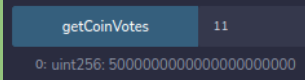
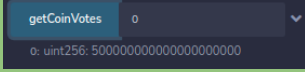
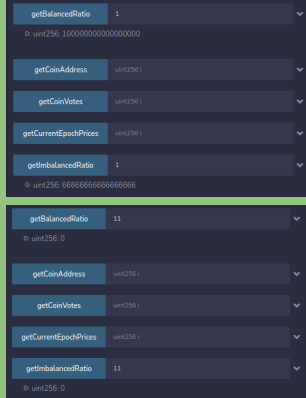
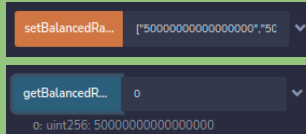


TEST ITEM	TESTER	TEST METHOD/IMPLEMENTATION	EXPECTED OUTPUT	ACTUAL OUTPUT	PROOF OF TEST COMPLETION	COMMENTS
Deployment Sequence (wallet allocations and pancake swap) - test 1	Amir + Phillip	Bruteforcing the sequence ensuring it is operable	N/A	N/A	Deployed variants are operable.	<p>A closer look into the process after accessors are built is needed.</p> <p>The sequence in question:</p> <ol style="list-style-type: none"> 1. change the router address to the appropriate address of the router to be utilized 2. set maxTx amount to infinite - or to a threshold? 3. deploy contract note - don't deploy Address.sol by accident 4. liquidity provision is manual through the DEX's website at this stage - the ratio of questions to provide is up to debate 5. send contract some of the chain's native token for gas 6. revoke ownership 7. if manual, transfer burn, vault, and bounty amounts 8. verify contract on explorer
ensure getTokenPrice works as expected - test 1	Grady	Cross referencing testnetBNB prices in accordance with testnet DEX (pancake.kiemtienonline360.com) with [WBNB, BUSD] pair	Method returns same prices (+- price impact, see below)	_getTokenPrice : 1WBNB = 639.984BUSD TestDEX: 1 WBNB =		<p>References:</p> <p>BSC testnet WBNB: https://testnet.bscscan.com/address/0xae13d989dac2f0debff460ac112a837c89bba7cd </p> <p>https://pancake.kiemtienonline360.com/#/swap (WBNB, BUSD pair) </p>

				640.626BUSD		
ensure getTokenPrice works as expected - test 2	Amir (Elmore, Phillip have verified before)	Cross-referencing testnet BNB prices on the testnet DEX vs output whilst accounting for price impact (~0.2-0.3%)	~[99.7-100.3]% of actual price	The price deviation is the price impact percentage exactly		<p>BSC testnet ETH: https://testnet.bscscan.com/address/0x8babb98678facc7342735486c851abd7a0d17ca</p> <p>BSC testnet BUSD: https://testnet.bscscan.com/token/0x78867bbeef44f2326bf8ddd1941a4439382ef2a7</p> <p>BSC testnet WBNB: https://testnet.bscscan.com/address/0xae13d989dac2f0debff460ac112a837c89baa7cd</p> <p>BSC testnet USDT: https://testnet.bscscan.com/address/0xf95a0fee0dd31b22626fa2e10ee6a223f8a684</p>
test rebalanceTransfer does not take any fees or reflections - test 1	Grady	Execute a transaction by calling smart contract rebalanceTransfer() method	No fees or reflections are taken	Token Balance is exactly the same both before and after rebalanceTrans fer() call	Verified through smart contract balanceOf() call and through testnet.bscscan.com	
test rebalanceTransfer does not take any fees or reflections - test 2	Amir	Conducting a vanilla transaction by calling smart contract method	No fees or reflections are taken	No fees or reflections are taken	Verified through smart contract balanceOf() call and through testnet.bscscan.com	

rebalanceReflect - extensive testing	Amir	A sequence of transactions has taken place to determine reflections are transferred as needed in proportion to users' wallet sizes from the burn wallet directly to users	Burn wallet amount transferred to users proportionately.	Burn wallet amount transferred to users proportionately.	https://testnet.bscscan.com/address/0x358bd3f30757c08f7659d183674fe78149e5738a	Examine the series of transactions and balances after these sample transfers are completed to reproduce the results of the test anew. Tiny drift (10^{-18} VSL prevails) - irreconcilable unless changing decimal precision further
test setCoinPrices* - test 1 *necessitates addresses to be preset	Amir + Grady	20 addresses are implicitly passed but as an address array.	Prices are to be updated in lastEpochPrices	Method executed - testing lastEpochPrices output	Using an accessor method, lastEpochPrices contains correct prices. 	The problem with the invalid ones (-32000 JSON RPC error) has been observed to be the lack of liquidity relative to BUSD
test updateAddresses - test 1	Amir+ Grady	20 addresses (testnet WBNB address) were passed in the format ["0x..", "0x..",..]	Addresses are expected to be updated to the input ones	Addresses are updated to the input ones	setCoinPrices() method runs successfully indicating that the _getTokenPrice() method accepts the addresses without halt.	Addresses must be input in the form ["0x..", "0x..",..] as JSON parsing clears everything up
test updateTheta	Amir	Called method with 2×10^{16} , 3×10^{16} (0.20, 0.3)	theta and theta_max are expected to update accordingly	correct updates		Had to temporarily set access modifier to public
test updateEpoch	Amir +	Called method with	epochLength to	Method	Resolved through	Update contract to pass uint

length	Grady	7 days	update accordingly	execution failed due to type incompatibility	changing argument to uint: 	argument not uint8; update contract to have epoch length of 6 days + 23 hours + 55 minutes
test vote() - test 1	Amir	Call updateEpoch().* Assign 100,000 tokens to Alice and 50,000 tokens to Bob, assign ratios 5% each for Alice and 10% for the first ten coins only for Bob. Attempt to vote TWICE in one epoch. Check the lastEpochVoteCast, totalVotesCast, and coinVotes variables	Alice and Bob can't vote again, totalVotesCast is 150,000. coinVotes[0..9] = 10,000 && coinVotes[10..19] = 5,000. Revoting impossible in the same epoch.	As expected - coinVotes[0..9] = 10,000 && coinVotes[10..19] = 5,000. Revoting impossible in the same epoch.	 	Serious error observed in computing maxVotesAllowed - rewritten with safemath div and mul calls - maxVotesAllowed = (rUsers.div(10**18)).mul(_tTotal).div(_rTotal.div(10**18)).div(1000);
test vote() - test 2	Grady	Call updateEpoch method. Give 10,000 tokens to test wallet to vote. Assign ratio of 5% across all tokens. Attempt to vote twice on one epoch.	Transaction executed. total VotesCast = 10000. coinVotes updated to take votes. Unable to vote again.	coinVotes[0..19] = 500. As expected. Cant vote again.		

test collectVotes() - test 1	Amir	Determine that the totalVotesCast gets reset to 0 and that lastVotesCast gets updated to 150000 from the prior example. Next ensure that balanced and imbalanced ratios are maintained correctly and experiment with different numbers for the next epoch.	We expect balancedRatio to be 10% for indices 0..9 and 0% for indices 10..19 and imbalanced ratio to be 6.66% for indices 0..9 and 0.00% (switched from 3.33%) for indices 10..19	The aforementioned results are what is yielded.		allocation error - calculation not divided by 10^{18} . (rectified)
test collectVotes() - test 2	Grady	Call collectVotes(). Determine that the totalVotesCast gets set to 0. Determine that lastVotesCast gets set to 10000.	Totalvotescast = 0 Lastvoteslast = 10,000 balancedRatio			<pre>imbalancedRatio[i] = (allocation < 499*10**14) ? 0 : allocation;</pre> <p>UPDATE LINE 1419 TO THIS CODE. [Bug reconciled]</p>
assign ratios - test 1	Amir	preset all balancedRatio votes to 5% each	Write a setter method, verify	All 20 ratios are set to be 5%		Initialize lastVotesCast with 1B votes for round 0. Potentially may, instead of setter, place this in the constructor instead?
assign ratios - test 2	Grady	Preset all balancedRatio votes to 5% each	Use setter method. Check getBalancedRatio method.	All 20 ratios are set to 5%		

assign initial timestamps etc.	Amir	set all timestamps accordingly for release and rebalancing	N/A	N/A	N/A	Have to modify the day scale to minutes for integration testing purposes down the line.
test logic of restartEpoch() and test code - test 1	Amir	determine all parameters are reset to 0 - coinVotes[], lastVotesCast, totalVotesCast, prices are renewed	Experiment by updating addresses and casting votes and determining all parameters get reset.	All the parameters specified in the updated function appropriately reset.	N/A.	<p>Modified code to clear multiple variables and reset others as well as to rectify some missing requirements for epoch restarts.</p> <p>Ensure to run updateAddresses first with valid addresses.</p> <pre>function restartEpoch() public only { for(uint8 i = 0; i < 20; i++){ coinVotes[i] = 0; balancedRatio[i] = 0; imbalancedRatio[i] = 0; } totalVotesCast = 0; lastVotesCast = 0; _setCoinPrices(); uint rUsers = _rTotal - _rOwned; maxVotesAllowed = (rUsers.div(2)); lastEpochRebalance = block.timestamp; }</pre>
test logic of restartEpoch() and test code - test 2	Grady	Add coin addresses, cast votes, call restart epoch, Determine all parameters are set to 0 - coinVotes[], lastVotesCast, totalVotesCast prices are updated to new determined values.	all parameters are set to 0 - coinVotes[], lastVotesCast, totalVotesCast. prices are updated to new determined values.	All the parameters specified prior are appropriately modified after the restartEpoch() method is called.		

Ensure the 20th index contains our own price in the implementation	Amir	Add a setter to manually update the price of VSL in lastEpochPrices[20] after liquidity relative to USD stablecoin is provided	<pre>function _setTokenPrice() public onlyOwner { lastEpochPrices[20] = _getTokenPrice(address(this)); }</pre>	This method is to be called strictly after liquidity is provided of VSL-USD to the exchange in question and must only be called once manually. It has been added to restartEpoch(). The reason it is not combined with _setCoinPrices() is for modularity of testing and clarity.
test logic of rebalanceEpoch() case 1 + 2a	Grady + Amir	<ul style="list-style-type: none"> -provide liquidity of BNB vs. Vessel -send 1B to burn, vault, and 100M to bounty through rebalanceTransfer -update addresses in the fund to link, eth, etc. -update the old prices using _setLastCoinPrice() and _setLastTokenPrice() -wait some time (to ensure a change in price occurs within the fund) [buy one or two of the wrapped tokens or sell manually instead] -call updateAllPrices() to set current prices -set some variable balancedRatios[] -call updateAllDeltas() -*remember theta - update it appropriately -determine which case is to be run -manually track the case methods perform as expected -call _rebalanceWrapup() - (vote twice in the billions - expect 20 million votes cast because of max threshold) 	<ul style="list-style-type: none"> -case 1: buy link -case 2a: sell link 	
rebalanceEpoch() [extended] case 2b test	Grady + Amir	<p>[Same setup as above with the same sequence]</p> <p>-concretely testing entering case 2b below:</p> <p>$u = 3.99b$, $v = 3.99b$, $b = 2.02b$</p> <p>CASE DETERMINATION: $\text{delta}_t = -1578341585830615$ $\text{delta}_w = -1711565902088480$</p>	<p>delta_2 tokens must be reflected from the burn wallet to the users:</p> <p>$\text{delta}_2 = 531565$</p>	

`delta_t > delta_w`

CASE 2:

`diff_t_w = 133224316257865`
`theta = 5000000000000000 (set to 5%)`
`delta = 133224316257865 * 3990000000000000000000000000 /`
`10**18 = 5.315650218688814 * 10**23`
`delta = 531565 * 10**18`


`531565 * 10**18 > u-v = 0 =>`


CASE 2B UNLOCKED


`delta_1 = u-v = 0`
`delta_2 = 531565 * 10**18 - 0 = 531565 * 10**18`

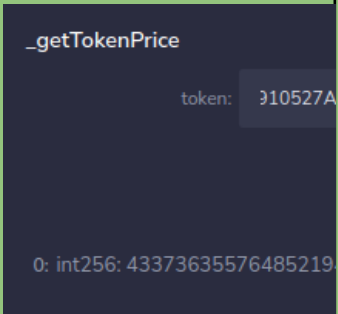
[nothing gets transferred from burn to vault]
`x = (b+v-u)/2 = 2.02b + 0 / 2 = 1.01b`
`f = 0`

`delta_2 < x-f, delta_2 = 531565 * 10**18`
`[x - f = x = 1.01*10**9*10**18 = 1.01*10**27]`

0... For 0  yacht (YCT5.2...)

0... For 531,565.02186888135  y

0... For 531,565.02186888135  y

rewrite _getTokenPrice to perform a conversion from token -> BNB/ETH/FTM/ etc. -> USD stablecoin to make the method far more stable - first derive price relative to BNB/ETH/FTM then multiply by current BNB/ETH/FTM price in USD	Amir	Test _getQuote() and _getTokenPrice() with all possible inputs including stablecoin and native chain asset	All prices are within expected bounds	All prices are OUTSIDE of expected bounds but this is normal behaviour		The innate assumption here is that all blue chip assets on the BSC network follow the simple route of asset->BNB->BUSD and it is a very reasonable assumption to hold as liquidity is far far better on the real network with real assets, so the deviations if the route is different should not be problematic either
Deployment Sequence Upon Completion (finalized code and accessors) - extensive testing	Everyone - TBD					
Optimize transaction costs and test	Everyone - TBD					
Deploy the exact equivalent of this contract and link to webapp on fantom as a final test	TBD					