

0.7

BarnBridge Process Quality Review

Score: 92%

Overview

This is a Process Quality Review completed on June 9th 2021. It was performed using the Process Review process (version 0.71) and is documented [here](#). The review was performed by Nic of DeFiSafety. Check out our [Telegram](#). The previous version of the review (0.6.2) is [here](#).

The final score of the review is 92%, a solid pass. The breakdown of the scoring is in [Scoring Appendix](#). For our purposes, a pass is 70%.

Summary of the Process

Very simply, the review looks for the following declarations from the developer's site. With these declarations, it is reasonable to trust the smart contracts.

- **Here are my smart contracts on the blockchain**
- **Here is the documentation that explains what my smart contracts do**
- **Here are the tests I ran to verify my smart contract**
- **Here are the audit(s) performed on my code by third party experts**
- **Here are the admin controls and strategies**

Disclaimer

This report is for informational purposes only and does not constitute investment advice of any kind, nor does it constitute an offer to provide investment advisory or other services. Nothing in this report shall be considered a solicitation or offer to buy or sell any security, token, future, option or other financial instrument or to offer or provide any investment advice or service to any person in any jurisdiction. Nothing contained in this report constitutes investment advice or offers any opinion with respect to the suitability of any security, and the views expressed in this report should not be taken as advice to buy, sell or hold any security. The information in this report should not be relied upon for the purpose of investing. In preparing the information contained in this report, we have not taken into account the investment needs, objectives and financial circumstances of any particular investor. This information has no regard to the specific investment objectives, financial situation and particular needs of any specific recipient of this information and investments discussed may not be suitable for all investors.

Any views expressed in this report by us were prepared based upon the information available to us at the time such views were written. The views expressed within this report are limited to DeFiSafety and the author and do not reflect those of any additional or third party and are strictly based upon DeFiSafety, its authors, interpretations and evaluation of relevant data. Changed or additional information could cause such

views to change. All information is subject to possible correction. Information may quickly become unreliable for various reasons, including changes in market conditions or economic circumstances.

This completed report is copyright (c) DeFiSafety 2021. Permission is given to copy in whole, retaining this copyright label.

Chain

This section indicates the blockchain used by this protocol.

Chain: Ethereum

Guidance:
Ethereum
Binance

Code and Team

This section looks at the code deployed on the Mainnet that gets reviewed and its corresponding software repository. The document explaining these questions is [here](#). This review will answer the questions;

- 1) Are the executing code addresses readily available? (%)
- 2) Is the code actively being used? (%)
- 3) Is there a public software repository? (Y/N)
- 4) Is there a development history visible? (%)
- 5) Is the team public (not anonymous)? (Y/N)

1) Are the executing code addresses readily available? (%)

Answer: 100%

They are available at website <https://integrations.barnbridge.com/smart-contract-addresses> as indicated in the [Appendix](#).

Guidance:

- | | |
|------|--|
| 100% | Clearly labelled and on website, docs or repo, quick to find |
| 70% | Clearly labelled and on website, docs or repo but takes a bit of looking |
| 40% | Addresses in mainnet.json, in discord or sub graph, etc |
| 20% | Address found but labelling not clear or easy to find |
| 0% | Executing addresses could not be found |

How to improve this score

Make the Ethereum addresses of the smart contract utilized by your application available on either your website or your GitHub (in the README for instance). Ensure the addresses is up to date. This is a very important question wrt to the final score.

2) Is the code actively being used? (%)

 Answer: 100%

Activity is 50 transactions a day on contract *barn.sol*, as indicated in the [Appendix](#).

Percentage Score Guidance

100%	More than 10 transactions a day
70%	More than 10 transactions a week
40%	More than 10 transactions a month
10%	Less than 10 transactions a month
0%	No activity

3) Is there a public software repository? (Y/N)

 Answer: Yes

GitHub: <https://github.com/BarnBridge>

Is there a public software repository with the code at a minimum, but normally test and scripts also (Y/N). Even if the repo was created just to hold the files and has just 1 transaction, it gets a Yes. For teams with private repos, this answer is No.

4) Is there a development history visible? (%)

 Answer: 100%

With 170 commits and 18 branches, this is a healthy repository.

This checks if the software repository demonstrates a strong steady history. This is normally demonstrated by commits, branches and releases in a software repository. A healthy history demonstrates a history of more than a month (at a minimum).

Guidance:

100%	Any one of 100+ commits, 10+branches
70%	Any one of 70+ commits, 7+branches
50%	Any one of 50+ commits, 5+branches

30%	Any one of 30+ commits, 3+branches
0%	Less than 2 branches or less than 10 commits

How to improve this score

Continue to test and perform other verification activities after deployment, including routine maintenance updating to new releases of testing and deployment tools. A public development history indicates clearly to the public the level of continued investment and activity by the developers on the application. This gives a level of security and faith in the application.

5) Is the team public (not anonymous)? (Y/N)

 Answer: Yes

The names of the team members can be found on their [about page](#).

For a yes in this question the real names of some team members must be public on the website or other documentation. If the team is anonymous and then this question is a No.

Documentation

This section looks at the software documentation. The document explaining these questions is [here](#).

Required questions are;

- 6) Is there a whitepaper? (Y/N)
- 7) Are the basic software functions documented? (Y/N)
- 8) Does the software function documentation fully (100%) cover the deployed contracts? (%)
- 9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)
- 10) Is it possible to trace from software documentation to the implementation in code (%)

6) Is there a whitepaper? (Y/N)

 Answer: Yes

Location: <https://github.com/BarnBridge/BarnBridge-Whitepaper>

How to improve this score

Ensure the white paper is available for download from your website or at least the software repository. Ideally update the whitepaper to meet the capabilities of your present application.

7) Are the basic software functions documented? (Y/N)

 Answer: Yes

Some of the Software functions for the BarnBridge_Barn are explained in their [github repository](#). Barnbridge additionally has a spec.md file for their BarnBridge-SmartYieldBonds contracts.

How to improve this score

Write the document based on the deployed code. For guidance, refer to the [SecurEth System Description Document](#).

8) Does the software function documentation fully (100%) cover the deployed contracts? (%)

 Answer: 80%

Extensive DAO and SMART Yield function and spec documentation in their developer [guide](#).

Extensive Barn function and spec documentation in their [Barn repository](#).

Other function/spec documentation can also be found in the [SMART Yield repository](#), and in the [DAO repository](#).

Extensive token function documentation in their [developer guide](#).

Function/spec documentation is lacking for the [YieldFarmContinuous repository](#).

Note: Overall improvement by making <https://integrations.barnbridge.com/> more publicly available.

Guidance:

- | | |
|-------|---|
| 100% | All contracts and functions documented |
| 80% | Only the major functions documented |
| 79-1% | Estimate of the level of software documentation |
| 0% | No software documentation |

How to improve this score

This score can improve by adding content to the requirements document such that it comprehensively covers the requirements. For guidance, refer to the [SecurEth System Description Document](#). Using tools that aid traceability detection will help.

9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)

 Answer: 26%

Code examples are in the [Appendix](#). As per the [SLOC](#), there is 26% commenting to code (CtC).

The Comments to Code (CtC) ratio is the primary metric for this score.

Guidance:

- 100% CtC > 100 Useful comments consistently on all code
- 90-70% CtC > 70 Useful comment on most code
- 60-20% CtC > 20 Some useful commenting
- 0% CtC < 20 No useful commenting

How to improve this score

This score can improve by adding comments to the deployed code such that it comprehensively covers the code. For guidance, refer to the [SecurEth Software Requirements](#).

10) Is it possible to trace from software documentation to the implementation in code (%)

 Answer: 60%

With <https://integrations.barnbridge.com/> being more publicly available and being implemented more in their GitHub repos,

Guidance:

- 100% Clear explicit traceability between code and documentation at a requirement level for all code
- 60% Clear association between code and documents via non explicit traceability
- 40% Documentation lists all the functions and describes their functions
- 0% No connection between documentation and code

How to improve this score

This score can improve by adding traceability from requirements to code such that it is clear where each requirement is coded. For reference, check the SecurEth guidelines on [traceability](#).

Testing

This section looks at the software testing available. It is explained in this [document](#). This section answers the following questions;

- 11) Full test suite (Covers all the deployed code) (%)
- 12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)
- 13) Scripts and instructions to run the tests (Y/N)
- 14) Report of the results (%)
- 15) Formal Verification test done (%)
- 16) Stress Testing environment (%)

11) Is there a Full test suite? (%)

 Answer: 100%

Code examples are in the [Appendix](#). As per the [SLOC](#), there is 185% testing to code (TtC).

This score is guided by the Test to Code ratio (TtC). Generally a good test to code ratio is over 100%. However the reviewers best judgement is the final deciding factor.

Guidance:

- 100% TtC > 120% Both unit and system test visible
- 80% TtC > 80% Both unit and system test visible
- 40% TtC < 80% Some tests visible
- 0% No tests obvious

How to improve this score

This score can improve by adding tests to fully cover the code. Document what is covered by traceability or test results in the software repository.

12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)

 Answer: 99%

There is no indication of code coverage in the deployed docs, but it is indicated in the [Quantstamp audit](#), and there are clearly a complete set of tests in the [Barn repository](#), [YieldFarmContinous Repository](#), and [SMART Yield repository](#).

Note: The DAO repository needs a more complete set of tests.

Guidance:

- 100% Documented full coverage
- 99-51% Value of test coverage from documented results
- 50% No indication of code coverage but clearly there is a reasonably complete set of tests
- 30% Some tests evident but not complete
- 0% No test for coverage seen

How to improve this score

This score can improve by adding tests achieving full code coverage. A clear report and scripts in the software repository will guarantee a high score.

13) Scripts and instructions to run the tests (Y/N)

 Answer: Yes

Scripts and instructions to run the tests can be found in their [GitHub repository](#).

How to improve this score

Add the scripts to the repository and ensure they work. Ask an outsider to create the environment and run the tests. Improve the scripts and docs based on their feedback.

14) Report of the results (%)

 Answer: 90%

Detailed test report in the [Barn repository](#), and [YieldFarmContinous Repository](#).

Detailed test report lacking for the SMART Yield repository and DAO repository.

Note: Unexplained misses

Guidance:

100% Detailed test report as described below

70% GitHub Code coverage report visible

0% No test report evident

How to improve this score

Add a report with the results. The test scripts should generate the report or elements of it.

15) Formal Verification test done (%)

 Answer: 0%

There is no evidence of any formal verification testing having been done.

16) Stress Testing environment (%)

 Answer: 100%

BarnBridge Testnet on Kovan.

- Kovan tokens available: BOND, USDC, DAI, USDT

- Use the faucets to get tokens
- <https://testnet.app.barnbridge.com/>
-

Security

This section looks at the 3rd party software audits done. It is explained in this [document](#). This section answers the following questions;

- 17) Did 3rd Party audits take place? (%)
- 18) Is the bounty value acceptably high?

17) Did 3rd Party audits take place? (%)

 Answer: 100%

Hacken did a review on September 29th, 2020. The is a Quantstamp audit also.

BarnBridge was first released on November 17th, 2020.

Most of the issues outline by the audits were fixed.

Guidance:

- 100% Multiple Audits performed before deployment and results public and implemented or not required
- 90% Single audit performed before deployment and results public and implemented or not required
- 70% Audit(s) performed after deployment and no changes required. Audit report is public
- 50% Audit(s) performed after deployment and changes needed but not implemented
- 20% No audit performed
- 0% Audit Performed after deployment, existence is public, report is not public and no improvements deployed OR smart contract address' not found, question

18) Is the bounty value acceptably high (%)

 Answer: 80%

A Bug Bounty with ImmuneFi was released with a value of 500,000\$.

Guidance:

- 100% Bounty is 10% TVL or at least \$1M AND active program (see below)
- 90% Bounty is 5% TVL or at least 500k AND active program

- 80% Bounty is 5% TVL or at least 500k
- 70% Bounty is 100k or over AND active program
- 50% Bounty is 100k or over
- 40% Bounty is 50k or over
- 20% Bug bounty program bounty is less than 50k
- 0% No bug bounty program offered

Active program means a third party actively driving hackers to the site. Inactive program would be static mention on the docs.

Access Controls

This section covers the documentation of special access controls for a DeFi protocol. The admin access controls are the contracts that allow updating contracts or coefficients in the protocol. Since these contracts can allow the protocol admins to "change the rules", complete disclosure of capabilities is vital for user's transparency. It is explained in this [document](#). The questions this section asks are as follow;

- 19) Can a user clearly and quickly find the status of the admin controls?
- 20) Is the information clear and complete?
- 21) Is the information in non-technical terms that pertain to the investments?
- 22) Is there Pause Control documentation including records of tests?

19) Can a user clearly and quickly find the status of the access controls (%)

 Answer: 100%

Access controls can be found in their [whitepaper](#) (3.2), as well as in their [GitBooks](#).

Guidance:

- 100% Clearly labelled and on website, docs or repo, quick to find
- 70% Clearly labelled and on website, docs or repo but takes a bit of looking
- 40% Access control docs in multiple places and not well labelled
- 20% Access control docs in multiple places and not labelled
- 0% Admin Control information could not be found

20) Is the information clear and complete (%)

 Answer: 90%

Capabilities for change are described in their [whitepaper](#) (3.2).

Contracts clearly labelled as upgradeable in 3.2.

Defined roles are described in 3.2.

Guidance:

All the contracts are immutable -- 100% OR

- a) All contracts are clearly labelled as upgradeable (or not) -- 30% AND
- b) The type of ownership is clearly indicated (OnlyOwner / MultiSig / Defined Roles) -- 30% AND
- c) The capabilities for change in the contracts are described -- 30%

How to improve this score

Create a document that covers the items described above. An [example](#) is enclosed.

21) Is the information in non-technical terms that pertain to the investments (%)

 Answer: 90%

Clear language in <https://docs.barnbridge.com/governance/beginners-guide-to-governance>.

Guidance:

- 100% All the contracts are immutable
- 90% Description relates to investments safety and updates in clear, complete non-software I language
- 30% Description all in software specific language
- 0% No admin control information could not be found

How to improve this score

Create a document that covers the items described above in plain language that investors can understand. An [example](#) is enclosed.

22) Is there Pause Control documentation including records of tests (%)

 Answer: 80%

<https://docs.barnbridge.com/access-controls>

Guidance:

- 100% All the contracts are immutable or no pause control needed and this is explained OR
- 100% Pause control(s) are clearly documented and there is records of at least one test within 3 months
- 80% Pause control(s) explained clearly but no evidence of regular tests
- 40% Pause controls mentioned with no detail on capability or tests
- 0% Pause control not documented or explained

How to improve this score

Create a document that covers the items described above in plain language that investors can understand. An [example](#) is enclosed.

Appendices

Author Details

The author of this review is Rex of DeFi Safety.

Email : rex@defisafety.com Twitter : @defisafety

I started with Ethereum just before the DAO and that was a wonderful education. It showed the importance of code quality. The second Parity hack also showed the importance of good process. Here my aviation background offers some value. Aerospace knows how to make reliable code using quality processes.

I was coaxed to go to EthDenver 2018 and there I started [SecuEth.org](#) with Bryant and Roman. We created guidelines on good processes for blockchain code development. We got [EthFoundation funding](#) to assist in their development.

Process Quality Reviews are an extension of the SecurEth guidelines that will further increase the quality processes in Solidity and Vyper development.

DeFiSafety is my full time gig and we are working on funding vehicles for a permanent staff.

Scoring Appendix

PQ Audit Scoring Matrix (v0.7)	Total	BarnBridge	
	Points	Answer	Points
	Total	260	238.45
Code and Team			
1) Are the executing code addresses readily available? (%)	20	100%	20
2) Is the code actively being used? (%)	5	100%	5
3) Is there a public software repository? (Y/N)	5	Y	5
4) Is there a development history visible? (%)	5	100%	5
5) Is the team public (not anonymous)? (Y/N)	15	Y	15
Code Documentation			
6) Is there a whitepaper? (Y/N)	5	Y	5
7) Are the basic software functions documented? (Y/N)	10	Y	10
8) Does the software function documentation fully (100%) cover the deployed contracts?	15	80%	12
9) Are there sufficiently detailed comments for all functions within the deployed contracts?	5	50%	2.5
10) Is it possible to trace from software documentation to the implementation in code (%)	10	60%	6
Testing			
11) Full test suite (Covers all the deployed code) (%)	20	100%	20
12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)	5	99%	4.95
13) Scripts and instructions to run the tests? (Y/N)	5	Y	5
14) Report of the results (%)	10	90%	9
15) Formal Verification test done (%)	5	0%	0
16) Stress Testing environment (%)	5	100%	5
Security			
17) Did 3rd Party audits take place? (%)	70	100%	70

18) Is the bug bounty acceptable high? (%)	10	80%	8
Access Controls			
19) Can a user clearly and quickly find the status of the admin controls	5	100%	5
20) Is the information clear and complete	10	90%	9
21) Is the information in non-technical terms	10	90%	9
22) Is there Pause Control documentation including records of tests	10	80%	8

Section Scoring			
Code and Team	50	100%	
Documentation	45	79%	
Testing	50	88%	
Security	80	98%	
Access Controls	35	89%	

Executing Code Appendix

BarnBridge DAO

The BarnBridge DAO is built on the Diamond Standard for smart contracts.

Contracts	Code	Address
BOND Token	GitHub	0x0391D2021f89DC339F60Fff84546EA23E337750f
Barn (Governance Staking)	GitHub	0x10e138877df69ca44fdc68655f86c88cde142d7f
DiamondCutFacet	GitHub	0x767f7d9E655161C9E6D8a3Db565666FCa2BDf4
DiamondLoupeFacet	GitHub	0x04499B879F6A7E75802cd09354eF2B788BF4Cf26
OwnershipFacet	GitHub	0xeB8E3e48F770C5c13D9De2203Fc307B6D04381FF
ChangeRewardsFacet	GitHub	0xb93E511D913A17826D2Df5AC8BE122C0EBd1A26d
BarnFacet	GitHub	0xA62dA56e9a330646386365dC6B2945b5C4d120ed
Launch-DAO (Deprecated)	-	0x5040a19E59FEcC9Bdf62CEa2F71a4604408C67BF

SMART Yield

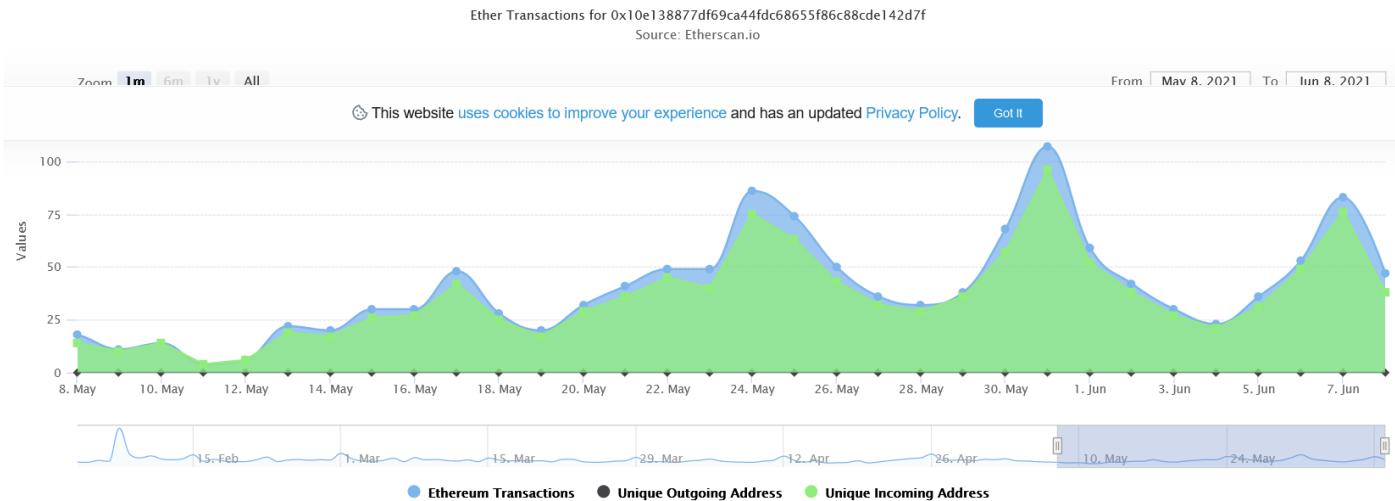
Contracts	Address
bb_cUSDC (Compound)	0x4b8d90d68f26def303dc6fc9b63a1aaec15840
bb_cDAI (Compound)	0x673f9488619821ab4f4155fdffe06f6139de518f
bb_aUSDC (Aave)	0x3cf46DA7D65E9aa2168a31b73dd4BeEA5cA1A1f1
bb_aUSDT (Aave)	0x660dAF6643191cF0eD045B861D820F283cA078fc

bb_aDAI (Aave)	0x6c9DaE2C40b1e5883847bF5129764e76Cb69Fc57
bb_aGUSD (Aave)	0x6324538cc222b43490dd95CEBF72cf09d98D9dAe
bb_crUSDC (Cream)	0x62e479060c89C48199FC7ad43b1432CC585BA1b9
bb_crUSDT (Cream)	0xc45F49bE156888a1C0C93dc0fE7dC89091E291f5
bb_crDAI (Cream)	0x89d82FdF095083Ded96B48FC6462Ed5dBD14151f

Yield Farming

Contracts	Code	Address
Pool 2 (Uniswap BOND/USDC LP)	GitHub	0xC25c37c387C5C909a94055F4f16184ca325D3a76
DAO Rewards Pool	GitHub	0x9d0cf50547d848cc4b6a12bedcf7696e9b334a22
cUSDC Rewards Pool	-	0x68Af34129755091E22F91899cEAC48657e5a5062
crUSDC Rewards Pool	-	0x7f7D4dFd9733ae12e6a5991d42aF16418f227b6E
crUSDT Rewards Pool	-	0xEA32E4E751D49757906E1153eF7A30fCAb1b6462
crDAI Rewards Pool	-	0x707c1bD52C4718BF040f350F7FE6ba0AdB484E8d
Pool 1 (Stablecoins - Deprecated)	GitHub	0xB3F7abF8FA1Df0fF61C5AC38d35e20490419f4bb

Code Used Appendix



Example Code Appendix

```
1 // SPDX-License-Identifier: Apache-2.0
2 pragma solidity 0.7.6;
3 pragma experimental ABIEncoderV2;
4
5 import "./interfaces/IDiamondCut.sol";
6 import "./interfaces/IDiamondLoupe.sol";
7 import "./libraries/LibDiamond.sol";
8 import "./libraries/LibOwnership.sol";
9 import "./libraries/LibDiamondStorage.sol";
10 import "./interfaces/IERC165.sol";
11 import "./interfaces/IERC173.sol";
12
13 contract Barn {
14     constructor(IDiamondCut.FacetCut[] memory _diamondCut, address _owner) payable {
15         require(_owner != address(0), "owner must not be 0x0");
16
17         LibDiamond.diamondCut(_diamondCut, address(0), new bytes(0));
18         LibOwnership.setContractOwner(_owner);
19
20         LibDiamondStorage.DiamondStorage storage ds = LibDiamondStorage.diamondStorage();
21
22         // adding ERC165 data
23         ds.supportedInterfaces[type(IERC165).interfaceId] = true;
24         ds.supportedInterfaces[type(IDiamondCut).interfaceId] = true;
25         ds.supportedInterfaces[type(IDiamondLoupe).interfaceId] = true;
26         ds.supportedInterfaces[type(IERC173).interfaceId] = true;
27     }
28
29     // Find facet for function that is called and execute the
30     // function if a facet is found and return any value.
31     fallback() external payable {
32         LibDiamondStorage.DiamondStorage storage ds = LibDiamondStorage.diamondStorage();
33
34         address facet = address(bytes20(ds.facets[msg.sig].facetAddress));
35         require(facet != address(0), "Diamond: Function does not exist");
36
37         assembly {
38             calldatacopy(0, 0, calldatasize())
39             let result := delegatecall(gas(), facet, 0, calldatasize(), 0, 0)
40             returndatacopy(0, 0, returndatasize())
41             switch result
42             case 0 {
43                 revert(0, returndatasize())
44             }
45             default {
46                 return (0, returndatasize())
47             }
48         }
49     }
50
51     receive() external payable {}
52 }
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	8	964	194	157	613	71

Comments to Code $157/613 = 26\%$

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	7	1482	324	24	1134	23

Tests to Code $1134/613 = 185\%$