

# CERTIK AUDIT REPORT FOR KSTARLIVE



Request Date: 2019-07-29  
Revision Date: 2019-07-31  
Platform Name: Luniverse



# Contents

<b>Disclaimer</b>	<b>1</b>
<b>About CertiK</b>	<b>2</b>
<b>Exective Summary</b>	<b>3</b>
<b>Vulnerability Classification</b>	<b>3</b>
<b>Testing Summary</b>	<b>4</b>
Audit Score . . . . .	4
Type of Issues . . . . .	4
Vulnerability Details . . . . .	5
<b>Manual Review Notes</b>	<b>6</b>
<b>Static Analysis Results</b>	<b>8</b>
<b>Formal Verification Results</b>	<b>10</b>
How to read . . . . .	10
<b>Source Code with CertiK Labels</b>	<b>55</b>

## Disclaimer

This Report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Verification Services Agreement between CertiK and KStarLive(the “Company”), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, the “Agreement”). This Report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This Report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK’s prior written consent.

## About CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, has developed a proprietary Formal Verification technology to apply rigorous and complete mathematical reasoning against code. This process ensures algorithms, protocols, and business functionalities are secured and working as intended across all platforms.

CertiK differs from traditional testing approaches by employing Formal Verification to mathematically prove blockchain ecosystem and smart contracts are hacker-resistant and bug-free. CertiK uses this industry-leading technology together with standardized test suites, static analysis, and expert manual review to create a full-stack solution for our partners across the blockchain world to secure 1.4B in assets.

For more information: <https://certik.org/>

## Executive Summary

This report has been prepared as the product of the Smart Contract Audit request by KStarLive. This audit was conducted to discover issues and vulnerabilities in the source code of KStarLive's Smart Contracts. Utilizing CertiK's Formal Verification Platform, Static Analysis, and Manual Review, a comprehensive examination has been performed. The auditing process pays special attention to the following considerations.

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessment of the codebase for best practice and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line by line manual review of the entire codebase by industry experts.

## Vulnerability Classification

For every issue found, CertiK categorizes them into 3 buckets based on its risk level:

### Critical

The code implementation does not match the specification, or it could result in loss of funds for contract owner or users.

### Medium

The code implementation does not match the specification at certain conditions, or it could affect the security standard by lost of access control.

### Low

The code implementation is not a best practice, or use a suboptimal design pattern, which may lead to security vulnerabilities, but no concern found yet.

## Testing Summary

# PASS

*CERTIK believes this smart contract passes security qualifications to be listed on digital asset exchanges.*

*Jul 31, 2019*



## Type of Issues

CertiK smart label engine applied 100% covered formal verification labels on the source code, and scanned the code using our proprietary static analysis and formal verification engine to detect the follow type of issues.

Title	Description	Issues	SWC ID
Integer Overflow and Underflow	An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum size of a type.	0	SWC-101
Function incorrectness	Function implementation does not meet the specification, leading to intentional or unintentional vulnerabilities.	0	
Buffer Overflow	An attacker is able to write to arbitrary storage locations of a contract if array of out bound happens	0	SWC-124
Reentrancy	A malicious contract can call back into the calling contract before the first invocation of the function is finished.	0	SWC-107
Transaction Order Dependence	A race condition vulnerability occurs when code depends on the order of the transactions submitted to it.	0	SWC-114
Timestamp Dependence	Timestamp can be influenced by minors to some degree.	0	SWC-116
Insecure Compiler Version	Using an fixed outdated compiler version or floating pragma can be problematic, if there are publicly disclosed bugs and issues that affect the current compiler version used.	1	SWC-102 SWC-103
Insecure Randomness	Block attributes are insecure to generate random numbers, as they can be influenced by minors to some degree.	0	SWC-120

“tx.origin” for authorization	tx.origin should not be used for authorization. Use msg.sender instead.	0	SWC-115
Delegatecall to Untrusted Callee	Calling into untrusted contracts is very dangerous, the target and arguments provided must be sanitized.	0	SWC-112
State Variable Default Visibility	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	0	SWC-108
Function Default Visibility	Functions are public by default. A malicious user is able to make unauthorized or unintended state changes if a developer forgot to set the visibility.	0	SWC-100
Uninitialized variables	Uninitialized local storage variables can point to other unexpected storage variables in the contract.	0	SWC-109
Assertion Failure	The assert() function is meant to assert invariants. Properly functioning code should never reach a failing assert statement.	0	SWC-110
Deprecated Solidity Features	Several functions and operators in Solidity are deprecated and should not be used as best practice.	0	SWC-111
Unused variables	Unused variables reduce code quality	0	

## Vulnerability Details

### Critical

No issue found.

### Medium

No issue found.

### Low

No issue found.

# Manual Review Notes

## Review Details

### Source Code SHA-256 Checksum

- **ERC20Token.sol**  
b89d2a3fd7d45c7f90e1949dbc0cb67cb9d62ae74bb67accc5924e6b3de8b19b
- **LinearMintableToken.sol**  
9641d0b8896dc78a524e99fc655cc16a74719f230144c6a49f802a258623ddde
- **MainToken.sol**  
f8221358c9820da03e7acf08c2f18fe7f023971b13bac367286b3bbec0d32af2
- **Ownable.sol**  
5db8b7271c1c283dca4be7b5fb81ebafd207f0cd1b377d041feadbb66862aeed
- **SafeMath.sol**  
30465ec63e5f02d79d09da2ddbf3f963ac128c130322dd123d41fbcfc6642de5
- **TokenRecipient.sol**  
a5af13920e2a00dd888d4c75000227fd20ad2e929596c164ae05a29655956fab

## Summary

CertiK was chosen by KStarLive to audit the design and implementation of its MainToken smart contract. To ensure comprehensive protection, the source code has been analyzed by the proprietary CertiK formal verification engine and manually reviewed by our smart contract experts and engineers. That end-to-end process ensures proof of stability as well as a hands-on, engineering-focused process to close potential loopholes and recommend design changes in accordance with the best practices in the space.

Overall we found the smart contracts to follow good practices. With the final update of source code and delivery of the audit report, we conclude that the contract is structurally sound and not vulnerable to any classically known anti-patterns or security issues. The audit report itself is not necessarily a guarantee of correctness or trustworthiness, and we always recommend to seek multiple opinions, keep improving the codebase, and more test coverage and sandbox deployments before the mainnet release.

## Recommendations

Items in this section are low impact to the overall aspects of the smart contracts, thus will let client to decide whether to have those reflected in the final deployed version of source codes. The entries are labeled **CRITICAL**, **MAJOR**, **MINOR**, **INFO**, **DISCUSSION** (in decreasing significance).

MainToken.sol

- **INFO** Recommend providing error messages to `require` statements.
- **INFO** Recommend locking the compiler version.



- **INFO** Recommend providing a getter method for `isLocked` for ease of use.

#### LinearMintableToken.sol

- **MINOR** `calculateMintAmount()`: Recommend using `SafeMath`.
- **INFO** Recommend providing error messages to `require` statements.
- **INFO** `mintInternal()`: Recommend changing `if ( mintingAmount == 0 )` to `require` () and supplement error messages.
- **INFO** Recommend locking the compiler version.

#### ERC20Token.sol

- **INFO** Recommend providing error messages to `require` statements.
- **INFO** Recommend locking the compiler version.

#### Ownable.sol

- **INFO** Recommend providing error messages to `require` statements.

# Static Analysis Results

## INSECURE\_COMPILER\_VERSION

Line 1 in File LinearMintableToken.sol

```
1 pragma solidity ^0.4.24;
```

! Version to compile has the following bug: 0.4.24: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor\_0.4.x, IncorrectEventSignatureInLibraries\_0.4.x, ABIEncoderV2PackedStorage\_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor\_0.4.x, IncorrectEventSignatureInLibraries\_0.4.x, ABIEncoderV2PackedStorage\_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

## TIMESTAMP\_DEPENDENCY

Line 54 in File LinearMintableToken.sol

```
54 createdTimestamp = block.timestamp;
```

! "block.timestamp" can be influenced by minors to some degree

## TIMESTAMP\_DEPENDENCY

Line 58 in File LinearMintableToken.sol

```
58 mintInternal(block.timestamp);
```

! "block.timestamp" can be influenced by minors to some degree

## INSECURE\_COMPILER\_VERSION

Line 1 in File Ownable.sol

```
1 pragma solidity ^0.4.24;
```

! Version to compile has the following bug: 0.4.24: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor\_0.4.x, IncorrectEventSignatureInLibraries\_0.4.x, ABIEncoderV2PackedStorage\_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor\_0.4.x, IncorrectEventSignatureInLibraries\_0.4.x, ABIEncoderV2PackedStorage\_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

## INSECURE\_COMPILER\_VERSION

Line 1 in File ERC20Token.sol

```
1 pragma solidity ^0.4.24;
```

! Version to compile has the following bug: 0.4.24: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor\_0.4.x, IncorrectEventSignatureInLibraries\_0.4.x, ABIEncoderV2PackedStorage\_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor\_0.4.x, IncorrectEventSignatureInLibraries\_0.4.x, ABIEncoderV2PackedStorage\_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

## INSECURE\_COMPILER\_VERSION

Line 1 in File MainToken.sol

```
1 pragma solidity ^0.4.24;
```

⚠ Version to compile has the following bug: 0.4.24: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor\_0.4.x, IncorrectEventSignatureInLibraries\_0.4.x, ABIEncoderV2PackedStorage\_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor\_0.4.x, IncorrectEventSignatureInLibraries\_0.4.x, ABIEncoderV2PackedStorage\_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

## INSECURE\_COMPILER\_VERSION

Line 1 in File SafeMath.sol

```
1 pragma solidity ^0.4.24;
```



⚠ Version to compile has the following bug: 0.4.24: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor\_0.4.x, IncorrectEventSignatureInLibraries\_0.4.x, ABIEncoderV2PackedStorage\_0.4.x, ExpExponentCleanup, EventStructWrongData 0.4.25: DynamicConstructorArgumentsClippedABIV2, UninitializedFunctionPointerInConstructor\_0.4.x, IncorrectEventSignatureInLibraries\_0.4.x, ABIEncoderV2PackedStorage\_0.4.x 0.4.26: DynamicConstructorArgumentsClippedABIV2

# Formal Verification Results

## How to read

### Detail for Request 1

transferFrom to same address


Verification date	 20, Oct 2018
Verification timespan	 395.38 ms

CERTIK label location	Line 30-34 in File howtoread.sol
-----------------------	----------------------------------

CERTIK label	30	/*@CTK FAIL "transferFrom to same address"
	31	@tag assume_completion
	32	@pre from == to
	33	@post __post.allowed[from][msg.sender] ==
	34	*/

Raw code location	Line 35-41 in File howtoread.sol
-------------------	----------------------------------

Raw code	35	function transferFrom(address from, address to
		) {
	36	balances[from] = balances[from].sub(tokens
	37	allowed[from][msg.sender] = allowed[from][
	38	balances[to] = balances[to].add(tokens);
	39	emit Transfer(from, to, tokens);
	40	return true;
	41	}

Counterexample	 This code violates the specification	
Initial environment	1	Counter Example:
	2	Before Execution:
	3	Input = {
	4	from = 0x0
	5	to = 0x0
	6	tokens = 0x6c
	7	}
	8	This = 0
	52	}
	53	balance: 0x0
	54	}
	55	}
Post environment	57	After Execution:
	58	Input = {
	59	from = 0x0
	60	to = 0x0
	61	tokens = 0x6c

## Formal Verification Request 1

If method completes, integer overflow would not happen.

📅 31, Jul 2019

🕒 156.82 ms

Line 20 in File LinearMintableToken.sol

20 `//@CTK NO_OVERFLOW`

Line 38-55 in File LinearMintableToken.sol

```
38 function registerLinearMint(  
39     uint256 _mintingSupply,  
40     uint256 _mintAmountPerPeriod,  
41     uint256 _intervalPeriodInDays  
42 ) external onlyOwner() {  
43     require(!mintingStatus);  
44     require(_mintingSupply > 0);  
45     require(totalSupply.add(_mintingSupply) <= maxSupply);  
46     require(_mintAmountPerPeriod > 0 );  
47     require(_intervalPeriodInDays > 0 );  
48  
49     mintingStatus = true;  
50  
51     mintingSupply = _mintingSupply;  
52     mintAmountPerPeriod = _mintAmountPerPeriod;  
53     intervalPeriodInDays = _intervalPeriodInDays;  
54     createdTimestamp = block.timestamp;  
55 }
```

✅ The code meets the specification.

## Formal Verification Request 2

Buffer overflow / array index out of bound would never happen.

📅 31, Jul 2019

🕒 76.84 ms

Line 21 in File LinearMintableToken.sol

21 `//@CTK NO_BUF_OVERFLOW`

Line 38-55 in File LinearMintableToken.sol

```
38 function registerLinearMint(  
39     uint256 _mintingSupply,  
40     uint256 _mintAmountPerPeriod,  
41     uint256 _intervalPeriodInDays  
42 ) external onlyOwner() {  
43     require(!mintingStatus);  
44     require(_mintingSupply > 0);  
45     require(totalSupply.add(_mintingSupply) <= maxSupply);  
46     require(_mintAmountPerPeriod > 0 );  
47     require(_intervalPeriodInDays > 0 );  
48  
49     mintingStatus = true;
```

```

50
51     mintingSupply = _mintingSupply;
52     mintAmountPerPeriod = _mintAmountPerPeriod;
53     intervalPeriodInDays = _intervalPeriodInDays;
54     createdTimestamp = block.timestamp;
55 }


```

✓ The code meets the specification.

## Formal Verification Request 3

Method will not encounter an assertion failure.

 31, Jul 2019

 43.46 ms

Line 22 in File LinearMintableToken.sol

```

22     // @CTK NO_ASF

```

Line 38-55 in File LinearMintableToken.sol

```

38     function registerLinearMint(
39         uint256 _mintingSupply,
40         uint256 _mintAmountPerPeriod,
41         uint256 _intervalPeriodInDays
42     ) external onlyOwner() {
43         require(!mintingStatus);
44         require(_mintingSupply > 0);
45         require(totalSupply.add(_mintingSupply) <= maxSupply);
46         require(_mintAmountPerPeriod > 0);
47         require(_intervalPeriodInDays > 0);
48
49         mintingStatus = true;
50
51         mintingSupply = _mintingSupply;
52         mintAmountPerPeriod = _mintAmountPerPeriod;
53         intervalPeriodInDays = _intervalPeriodInDays;
54         createdTimestamp = block.timestamp;
55     }


```

✓ The code meets the specification.

## Formal Verification Request 4

registerLinearMint correctness

 31, Jul 2019

 182.87 ms

Line 23-37 in File LinearMintableToken.sol

```

23     /* @CTK "registerLinearMint correctness"
24         @tag assume_completion
25         @post mintingStatus == false
26         @post totalSupply + _mintingSupply <= maxSupply

```

```

27  @post _owner == msg.sender
28  @post __post.mintingStatus == true
29  @post __post.mintingSupply == _mintingSupply
30  @post __post.mintingSupply > 0
31  @post __post.mintingSupply == _mintingSupply
32  @post __post.mintAmountPerPeriod == _mintAmountPerPeriod
33  @post __post.mintAmountPerPeriod > 0
34  @post __post.intervalPeriodInDays == _intervalPeriodInDays
35  @post __post.intervalPeriodInDays > 0
36  @post __post.createdTimestamp == block.timestamp
37  */

```

Line 38-55 in File LinearMintableToken.sol

```

38  function registerLinearMint(
39      uint256 _mintingSupply,
40      uint256 _mintAmountPerPeriod,
41      uint256 _intervalPeriodInDays
42  ) external onlyOwner() {
43      require(!mintingStatus);
44      require(_mintingSupply > 0);
45      require(totalSupply.add(_mintingSupply) <= maxSupply);
46      require(_mintAmountPerPeriod > 0 );
47      require(_intervalPeriodInDays > 0 );
48
49      mintingStatus = true;
50
51      mintingSupply = _mintingSupply;
52      mintAmountPerPeriod = _mintAmountPerPeriod;
53      intervalPeriodInDays = _intervalPeriodInDays;
54      createdTimestamp = block.timestamp;
55  }


```

✓ The code meets the specification.

## Formal Verification Request 5

If method completes, integer overflow would not happen.

 31, Jul 2019

 11075.7 ms

Line 61 in File LinearMintableToken.sol

```

61  //@CTK FAIL NO_OVERFLOW

```

Line 70-92 in File LinearMintableToken.sol

```

70  function mintInternal(uint256 blockTimestamp) internal {
71      require(mintingStatus);
72
73      address tokenOwner = owner();
74      uint256 mintingAmount = calculateMintAmount(blockTimestamp);
75      mintingAmount = mintingAmount.sub(mintedAmount);
76
77      if ( mintingAmount == 0 )
78          return;
79

```

```

80  if ( mintingAmount >= mintingSupply ) {
81      mintingAmount = mintingSupply.sub(mintedAmount);
82      mintingStatus = false;
83  }
84
85  totalSupply = totalSupply.add(mintingAmount);
86
87  _balances[tokenOwner] = _balances[tokenOwner].add(mintingAmount);
88
89  mintedAmount = mintedAmount.add(mintingAmount);
90
91  emit Minted(tokenOwner, mintingAmount, mintedAmount);
92  }

```

✗ This code violates the specification.

```

1  Counter Example:
2  Before Execution:
3      Input = {
4          blockTimestamp = 32
5      }
6      This = 0
7      Internal = {
8          __has_assertion_failure = false
9          __has_buf_overflow = false
10         __has_overflow = false
11         __has_returned = false
12         __reverted = false
13         msg = {
14             "gas": 0,
15             "sender": 0,
16             "value": 0
17         }
18     }
19     Other = {
20         block = {
21             "number": 0,
22             "timestamp": 0
23         }
24     }
25     Address_Map = [
26         {
27             "key": 0,
28             "value": {
29                 "contract_name": "LinearMintableToken",
30                 "balance": 0,
31                 "contract": {
32                     "SECONDS_IN_A_DAY": 1,
33                     "mintingStatus": true,
34                     "mintingSupply": 8,
35                     "intervalPeriodInDays": 1,
36                     "intervalCount": 0,
37                     "mintAmountPerPeriod": 129,
38                     "createdTimestamp": 15,
39                     "mintedAmount": 18,
40                     "_owner": 0,
41                     "name": "",
42                     "symbol": "",
43                     "totalSupply": 20,

```



```

44     "maxSupply": 0,
45     "decimals": 0,
46     "_balances": [
47         {
48             "key": 8,
49             "value": 2
50         },
51         {
52             "key": 1,
53             "value": 0
54         },
55         {
56             "key": 33,
57             "value": 1
58         },
59         {
60             "key": 0,
61             "value": 0
62         },
63         {
64             "key": 129,
65             "value": 0
66         },
67         {
68             "key": 128,
69             "value": 4
70         },
71         {
72             "key": "ALL_OTHERS",
73             "value": 16
74         }
75     ],
76     "_allowed": [
77         {
78             "key": 0,
79             "value": [
80                 {
81                     "key": 0,
82                     "value": 1
83                 },
84                 {
85                     "key": 4,
86                     "value": 0
87                 },
88                 {
89                     "key": "ALL_OTHERS",
90                     "value": 64
91                 }
92             ]
93         },
94         {
95             "key": 32,
96             "value": [
97                 {
98                     "key": 0,
99                     "value": 32
100                 },
101                 {

```

```

102         "key": "ALL_OTHERS",
103         "value": 64
104     }
105 ]
106 },
107 {
108     "key": "ALL_OTHERS",
109     "value": [
110         {
111             "key": "ALL_OTHERS",
112             "value": 16
113         }
114     ]
115 }
116 ]
117 }
118 }
119 },
120 {
121     "key": "ALL_OTHERS",
122     "value": "EmptyAddress"
123 }
124 ]
125
126 After Execution:
127 Input = {
128     blockTimestamp = 32
129 }
130 This = 0
131 Internal = {
132     __has_assertion_failure = false
133     __has_buf_overflow = false
134     __has_overflow = true
135     __has_returned = true
136     __reverted = false
137     msg = {
138         "gas": 0,
139         "sender": 0,
140         "value": 0
141     }
142 }
143 Other = {
144     block = {
145         "number": 0,
146         "timestamp": 0
147     }
148     mintingAmount = 0
149     tokenOwner = 0
150 }
151 Address_Map = [
152     {
153         "key": 0,
154         "value": {
155             "contract_name": "LinearMintableToken",
156             "balance": 0,
157             "contract": {
158                 "SECONDS_IN_A_DAY": 1,
159                 "mintingStatus": true,

```

```

160     "mintingSupply": 8,
161     "intervalPeriodInDays": 1,
162     "intervalCount": 0,
163     "mintAmountPerPeriod": 129,
164     "createdTimestamp": 15,
165     "mintedAmount": 18,
166     "_owner": 0,
167     "name": "",
168     "symbol": "",
169     "totalSupply": 20,
170     "maxSupply": 0,
171     "decimals": 0,
172     "_balances": [
173     {
174         "key": 8,
175         "value": 2
176     },
177     {
178         "key": 1,
179         "value": 0
180     },
181     {
182         "key": 33,
183         "value": 1
184     },
185     {
186         "key": 0,
187         "value": 0
188     },
189     {
190         "key": 129,
191         "value": 0
192     },
193     {
194         "key": 128,
195         "value": 4
196     },
197     {
198         "key": "ALL_OTHERS",
199         "value": 16
200     }
201 ],
202     "_allowed": [
203     {
204         "key": 0,
205         "value": [
206         {
207             "key": 0,
208             "value": 1
209         },
210         {
211             "key": 4,
212             "value": 0
213         },
214         {
215             "key": "ALL_OTHERS",
216             "value": 64
217         }

```

```


218         ]
219     },
220     {
221         "key": 32,
222         "value": [
223             {
224                 "key": 0,
225                 "value": 32
226             },
227             {
228                 "key": "ALL_OTHERS",
229                 "value": 64
230             }
231         ]
232     },
233     {
234         "key": "ALL_OTHERS",
235         "value": [
236             {
237                 "key": "ALL_OTHERS",
238                 "value": 16
239             }
240         ]
241     }
242 ]
243 }
244 }
245 },
246 {
247     "key": "ALL_OTHERS",
248     "value": "EmptyAddress"
249 }
250 ]

```

## Formal Verification Request 6

Buffer overflow / array index out of bound would never happen.

 31, Jul 2019

 191.55 ms

Line 62 in File LinearMintableToken.sol

```
62  // @CTK NO_BUF_OVERFLOW
```

Line 70-92 in File LinearMintableToken.sol

```

70  function mintInternal(uint256 blockTimestamp) internal {
71      require(mintingStatus);
72
73      address tokenOwner = owner();
74      uint256 mintingAmount = calculateMintAmount(blockTimestamp);
75      mintingAmount = mintingAmount.sub(mintedAmount);
76
77      if ( mintingAmount == 0 )
78          return;
79
80      if ( mintingAmount >= mintingSupply ) {

```

```

81     mintingAmount = mintingSupply.sub(mintedAmount);
82     mintingStatus = false;
83 }
84
85     totalSupply = totalSupply.add(mintingAmount);
86
87     _balances[tokenOwner] = _balances[tokenOwner].add(mintingAmount);
88
89     mintedAmount = mintedAmount.add(mintingAmount);
90
91     emit Minted(tokenOwner, mintingAmount, mintedAmount);
92 }


```

✓ The code meets the specification.

## Formal Verification Request 7

mintInternal NO\_ASF and correctness

 31, Jul 2019

 205.09 ms

Line 63-69 in File LinearMintableToken.sol

```

63  /*@CTK "mintInternal NO_ASF and correctness"
64     @tag assume_completion
65     @pre intervalPeriodInDays > 0
66     @pre SECONDS_IN_A_DAY > 0
67     @post mintingStatus == true
68     @post !(__has_assertion_failure)
69  */

```

Line 70-92 in File LinearMintableToken.sol

```

70  function mintInternal(uint256 blockTimestamp) internal {
71      require(mintingStatus);
72
73      address tokenOwner = owner();
74      uint256 mintingAmount = calculateMintAmount(blockTimestamp);
75      mintingAmount = mintingAmount.sub(mintedAmount);
76
77      if ( mintingAmount == 0 )
78          return;
79
80      if ( mintingAmount >= mintingSupply ) {
81          mintingAmount = mintingSupply.sub(mintedAmount);
82          mintingStatus = false;
83      }
84
85      totalSupply = totalSupply.add(mintingAmount);
86
87      _balances[tokenOwner] = _balances[tokenOwner].add(mintingAmount);
88
89      mintedAmount = mintedAmount.add(mintingAmount);
90
91      emit Minted(tokenOwner, mintingAmount, mintedAmount);
92  }

```

✓ The code meets the specification.

## Formal Verification Request 8

If method completes, integer overflow would not happen.

📅 31, Jul 2019

🕒 21.77 ms

Line 94 in File LinearMintableToken.sol

94 `//@CTK FAIL NO_OVERFLOW`

Line 102-107 in File LinearMintableToken.sol

```
102 function calculateMintAmount(uint256 blockTimestamp) public view returns (uint256
    mintAmount) {
103     uint256 pastDays = blockTimestamp.sub(createdTimestamp).div(SECONDS_IN_A_DAY);
104     uint256 pastIntervals = pastDays / intervalPeriodInDays + 1;
105
106     return pastIntervals * mintAmountPerPeriod;
107 }
```

✗ This code violates the specification.

```
1 Counter Example:
2 Before Execution:
3   Input = {
4     blockTimestamp = 128
5   }
6   This = 0
7   Internal = {
8     __has_assertion_failure = false
9     __has_buf_overflow = false
10    __has_overflow = false
11    __has_returned = false
12    __reverted = false
13    msg = {
14      "gas": 0,
15      "sender": 0,
16      "value": 0
17    }
18  }
19  Other = {
20    block = {
21      "number": 0,
22      "timestamp": 0
23    }
24    mintAmount = 0
25  }
26  Address_Map = [
27    {
28      "key": "ALL_OTHERS",
29      "value": {
30        "contract_name": "LinearMintableToken",
31        "balance": 0,
32        "contract": {
33          "SECONDS_IN_A_DAY": 40,
```

```

34         "mintingStatus": false,
35         "mintingSupply": 0,
36         "intervalPeriodInDays": 1,
37         "intervalCount": 0,
38         "mintAmountPerPeriod": 129,
39         "createdTimestamp": 64,
40         "mintedAmount": 0,
41         "_owner": 0,
42         "name": "",
43         "symbol": "",
44         "totalSupply": 0,
45         "maxSupply": 0,
46         "decimals": 0,
47         "_balances": [
48             {
49                 "key": 32,
50                 "value": 0
51             },
52             {
53                 "key": "ALL_OTHERS",
54                 "value": 4
55             }
56         ],
57         "_allowed": [
58             {
59                 "key": "ALL_OTHERS",
60                 "value": [
61                     {
62                         "key": 32,
63                         "value": 0
64                     },
65                     {
66                         "key": "ALL_OTHERS",
67                         "value": 4
68                     }
69                 ]
70             }
71         ]
72     }
73 }
74 ]
75 ]
76 ]

```

After Execution:

```

77     Input = {
78         blockTimestamp = 128
79     }
80     This = 0
81     Internal = {
82         __has_assertion_failure = false
83         __has_buf_overflow = false
84         __has_overflow = true
85         __has_returned = true
86         __reverted = false
87     }
88     msg = {
89         "gas": 0,
90         "sender": 0,
91         "value": 0

```

```

92     }
93 }
94 Other = {
95     block = {
96         "number": 0,
97         "timestamp": 0
98     }
99     mintAmount = 2
100     pastDays = 1
101     pastIntervals = 2
102 }
103 Address_Map = [
104     {
105         "key": "ALL_OTHERS",
106         "value": {
107             "contract_name": "LinearMintableToken",
108             "balance": 0,
109             "contract": {
110                 "SECONDS_IN_A_DAY": 40,
111                 "mintingStatus": false,
112                 "mintingSupply": 0,
113                 "intervalPeriodInDays": 1,
114                 "intervalCount": 0,
115                 "mintAmountPerPeriod": 129,
116                 "createdTimestamp": 64,
117                 "mintedAmount": 0,
118                 "_owner": 0,
119                 "name": "",
120                 "symbol": "",
121                 "totalSupply": 0,
122                 "maxSupply": 0,
123                 "decimals": 0,
124                 "_balances": [
125                     {
126                         "key": 32,
127                         "value": 0
128                     },
129                     {
130                         "key": "ALL_OTHERS",
131                         "value": 4
132                     }
133                 ],
134                 "_allowed": [
135                     {
136                         "key": "ALL_OTHERS",
137                         "value": [
138                             {
139                                 "key": 32,
140                                 "value": 0
141                             },
142                             {
143                                 "key": "ALL_OTHERS",
144                                 "value": 4
145                             }
146                         ]
147                     }
148                 ]
149             }

```




```
150     }
151   }
152 ]
```

## Formal Verification Request 9

Buffer overflow / array index out of bound would never happen.

 31, Jul 2019

 0.84 ms

Line 95 in File LinearMintableToken.sol

```
95  //@CTK NO_BUF_OVERFLOW
```

Line 102-107 in File LinearMintableToken.sol


```
102  function calculateMintAmount(uint256 blockTimestamp) public view returns (uint256
      mintAmount) {
103      uint256 pastDays = blockTimestamp.sub(createdTimestamp).div(SECONDS_IN_A_DAY);
104      uint256 pastIntervals = pastDays / intervalPeriodInDays + 1;
105
106      return pastIntervals * mintAmountPerPeriod;
107  }
```

 The code meets the specification.

## Formal Verification Request 10

calculateMintAmount NO\_ASF

 31, Jul 2019

 4.65 ms

Line 96-101 in File LinearMintableToken.sol

```
96  /*@CTK "calculateMintAmount NO_ASF"
97      @tag assume_completion
98      @pre intervalPeriodInDays > 0
99      @pre SECONDS_IN_A_DAY > 0
100      @post !(__has_assertion_failure)
101  */
```

Line 102-107 in File LinearMintableToken.sol

```
102  function calculateMintAmount(uint256 blockTimestamp) public view returns (uint256
      mintAmount) {
103      uint256 pastDays = blockTimestamp.sub(createdTimestamp).div(SECONDS_IN_A_DAY);
104      uint256 pastIntervals = pastDays / intervalPeriodInDays + 1;
105
106      return pastIntervals * mintAmountPerPeriod;
107  }
```

 The code meets the specification.

## Formal Verification Request 11

### Ownable

📅 31, Jul 2019

🕒 5.63 ms

Line 17-19 in File Ownable.sol

```
17  /*@CTK Ownable
18     @post __post._owner == msg.sender
19  */
```

Line 20-23 in File Ownable.sol

```
20  constructor () internal {
21      _owner = msg.sender;
22      emit OwnershipTransferred(address(0), _owner);
23  }
```

✅ The code meets the specification.

## Formal Verification Request 12

### owner

📅 31, Jul 2019

🕒 5.17 ms

Line 28-30 in File Ownable.sol

```
28  /*@CTK owner
29     @post __return == _owner
30  */
```

Line 31-33 in File Ownable.sol

```
31  function owner() public view returns (address) {
32      return _owner;
33  }
```

✅ The code meets the specification.

## Formal Verification Request 13

### isOwner

📅 31, Jul 2019

🕒 6.2 ms

Line 46-48 in File Ownable.sol

```
46  /*@CTK isOwner
47     @post __return == (msg.sender == _owner)
48  */
```

Line 49-51 in File Ownable.sol


```
49 function isOwner() public view returns (bool) {  
50     return msg.sender == _owner;  
51 }
```

✓ The code meets the specification.

## Formal Verification Request 14

### transferOwnership

 31, Jul 2019

 53.14 ms

Line 69-72 in File Ownable.sol

```
69 /*@CTK transferOwnership  
70     @tag assume_completion  
71     @post _owner == msg.sender  
72 */
```

Line 73-75 in File Ownable.sol


```
73 function transferOwnership(address newOwner) public onlyOwner {  
74     _transferOwnership(newOwner);  
75 }
```

✓ The code meets the specification.

## Formal Verification Request 15

### \_transferOwnership

 31, Jul 2019

 1.07 ms

Line 81-85 in File Ownable.sol

```
81 /*@CTK _transferOwnership  
82     @tag assume_completion  
83     @post newOwner != address(0)  
84     @post __post._owner == newOwner  
85 */
```

Line 86-90 in File Ownable.sol

```
86 function _transferOwnership(address newOwner) internal {  
87     require(newOwner != address(0));  
88     emit OwnershipTransferred(_owner, newOwner);  
89     _owner = newOwner;  
90 }
```

✓ The code meets the specification.

## Formal Verification Request 16

If method completes, integer overflow would not happen.

📅 31, Jul 2019

🕒 28.09 ms

Line 23 in File ERC20Token.sol

23 `//@CTK NO_OVERFLOW`

Line 32-42 in File ERC20Token.sol

```
32  constructor(string _name, string _symbol, uint8 _decimals, uint256 _initialSupply,
    uint256 _maxSupply) public {
33      require(_maxSupply >= _initialSupply);
34
35      name = _name;
36      symbol = _symbol;
37      decimals = _decimals;
38
39      _balances[msg.sender] = _initialSupply;
40      totalSupply = _initialSupply;
41      maxSupply = _maxSupply;
42  }
```

✅ The code meets the specification.

## Formal Verification Request 17

Buffer overflow / array index out of bound would never happen.

📅 31, Jul 2019

🕒 0.49 ms

Line 24 in File ERC20Token.sol

24 `//@CTK NO_BUF_OVERFLOW`

Line 32-42 in File ERC20Token.sol

```
32  constructor(string _name, string _symbol, uint8 _decimals, uint256 _initialSupply,
    uint256 _maxSupply) public {
33      require(_maxSupply >= _initialSupply);
34
35      name = _name;
36      symbol = _symbol;
37      decimals = _decimals;
38
39      _balances[msg.sender] = _initialSupply;
40      totalSupply = _initialSupply;
41      maxSupply = _maxSupply;
42  }
```

✅ The code meets the specification.

## Formal Verification Request 18

Method will not encounter an assertion failure.

📅 31, Jul 2019

🕒 0.48 ms

Line 25 in File ERC20Token.sol

```
25  // @CTK NO_ASF
```

Line 32-42 in File ERC20Token.sol

```
32  constructor(string _name, string _symbol, uint8 _decimals, uint256 _initialSupply,
33             uint256 _maxSupply) public {
34      require(_maxSupply >= _initialSupply);
35
36      name = _name;
37      symbol = _symbol;
38      decimals = _decimals;
39
40      _balances[msg.sender] = _initialSupply;
41      totalSupply = _initialSupply;
42      maxSupply = _maxSupply;
43  }
```

✅ The code meets the specification.

## Formal Verification Request 19

constructor correctness

📅 31, Jul 2019

🕒 4.65 ms

Line 26-31 in File ERC20Token.sol

```
26  /* @CTK "constructor correctness"
27     @tag assume_completion
28     @post __post._balances[msg.sender] == __post.totalSupply
29     @post __post._balances[msg.sender] == _initialSupply
30     @post __post.maxSupply == _maxSupply
31  */
```

Line 32-42 in File ERC20Token.sol

```
32  constructor(string _name, string _symbol, uint8 _decimals, uint256 _initialSupply,
33             uint256 _maxSupply) public {
34      require(_maxSupply >= _initialSupply);
35
36      name = _name;
37      symbol = _symbol;
38      decimals = _decimals;
39
40      _balances[msg.sender] = _initialSupply;
41      totalSupply = _initialSupply;
42      maxSupply = _maxSupply;
43  }
```

✓ The code meets the specification.

## Formal Verification Request 20

If method completes, integer overflow would not happen.

📅 31, Jul 2019

🕒 5.71 ms

Line 44 in File ERC20Token.sol

44 `//@CTK NO_OVERFLOW`

Line 50-52 in File ERC20Token.sol

```
50 function balanceOf(address _owner) public view returns (uint balance) {  
51 return _balances[_owner];  
52 }
```

✓ The code meets the specification.

## Formal Verification Request 21

Buffer overflow / array index out of bound would never happen.

📅 31, Jul 2019

🕒 0.35 ms

Line 45 in File ERC20Token.sol

45 `//@CTK NO_BUF_OVERFLOW`

Line 50-52 in File ERC20Token.sol

```
50 function balanceOf(address _owner) public view returns (uint balance) {  
51 return _balances[_owner];  
52 }
```

✓ The code meets the specification.

## Formal Verification Request 22

Method will not encounter an assertion failure.

📅 31, Jul 2019

🕒 0.33 ms

Line 46 in File ERC20Token.sol

46 `//@CTK NO_ASF`

Line 50-52 in File ERC20Token.sol


```
50 function balanceOf(address _owner) public view returns (uint balance) {  
51 return _balances[_owner];  
52 }
```

✓ The code meets the specification.

## Formal Verification Request 23

balanceOf correctness

 31, Jul 2019

 0.34 ms

Line 47-49 in File ERC20Token.sol

```
47  /*@CTK "balanceOf correctness"
48     @post balance == _balances[_owner]
49  */
```

Line 50-52 in File ERC20Token.sol


```
50  function balanceOf(address _owner) public view returns (uint balance) {
51      return _balances[_owner];
52  }
```

 The code meets the specification.

## Formal Verification Request 24

If method completes, integer overflow would not happen.

 31, Jul 2019

 6.21 ms

Line 54 in File ERC20Token.sol

```
54  //@CTK NO_OVERFLOW
```

Line 60-62 in File ERC20Token.sol


```
60  function allowance(address _owner, address _spender) public view returns (uint256
      remaining) {
61      return _allowed[_owner][_spender];
62  }
```

 The code meets the specification.

## Formal Verification Request 25

Buffer overflow / array index out of bound would never happen.

 31, Jul 2019

 0.33 ms

Line 55 in File ERC20Token.sol

```
55  //@CTK NO_BUF_OVERFLOW
```

Line 60-62 in File ERC20Token.sol

```
60  function allowance(address _owner, address _spender) public view returns (uint256
      remaining) {
61      return _allowed[_owner][_spender];
62  }
```

 The code meets the specification.

## Formal Verification Request 26

Method will not encounter an assertion failure.

📅 31, Jul 2019

🕒 0.33 ms

Line 56 in File ERC20Token.sol

```
56 // @CTK NO_ASF
```

Line 60-62 in File ERC20Token.sol

```
60 function allowance(address _owner, address _spender) public view returns (uint256
    remaining) {
61     return _allowed[_owner][_spender];
62 }
```

✅ The code meets the specification.

## Formal Verification Request 27

allowance correctness

📅 31, Jul 2019

🕒 0.34 ms

Line 57-59 in File ERC20Token.sol

```
57 /* @CTK "allowance correctness"
58     @post remaining == _allowed[_owner][_spender]
59 */
```

Line 60-62 in File ERC20Token.sol

```
60 function allowance(address _owner, address _spender) public view returns (uint256
    remaining) {
61     return _allowed[_owner][_spender];
62 }
```

✅ The code meets the specification.

## Formal Verification Request 28

If method completes, integer overflow would not happen.

📅 31, Jul 2019

🕒 182.53 ms

Line 69 in File ERC20Token.sol

```
69 // @CTK NO_OVERFLOW
```

Line 81-84 in File ERC20Token.sol

```
81 function transfer(address _to, uint256 _value) public returns (bool success) {
82     _transfer(msg.sender, _to, _value);
83     return true;
84 }
```



✓ The code meets the specification.

## Formal Verification Request 29

Buffer overflow / array index out of bound would never happen.

📅 31, Jul 2019

🕒 11.26 ms

Line 70 in File ERC20Token.sol

```
70 // @CTK NO_BUF_OVERFLOW
```

Line 81-84 in File ERC20Token.sol

```
81 function transfer(address _to, uint256 _value) public returns (bool success) {  
82     _transfer(msg.sender, _to, _value);  
83     return true;  
84 }
```

✓ The code meets the specification.

## Formal Verification Request 30

Method will not encounter an assertion failure.

📅 31, Jul 2019

🕒 12.48 ms

Line 71 in File ERC20Token.sol

```
71 // @CTK NO_ASF
```

Line 81-84 in File ERC20Token.sol

```
81 function transfer(address _to, uint256 _value) public returns (bool success) {  
82     _transfer(msg.sender, _to, _value);  
83     return true;  
84 }
```

✓ The code meets the specification.

## Formal Verification Request 31

transfer correctness

📅 31, Jul 2019

🕒 81.95 ms

Line 72-80 in File ERC20Token.sol

```
72 /* @CTK "transfer correctness"  
73     @tag assume_completion  
74     @post _to != 0x0  
75     @post _value <= _balances[msg.sender]
```

```

76  @post _to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender] -
    _value
77  @post _to != msg.sender -> __post._balances[_to] == _balances[_to] + _value
78  @post _to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
79  @post success == true
80  */

```

Line 81-84 in File ERC20Token.sol

```

81  function transfer(address _to, uint256 _value) public returns (bool success) {
82    _transfer(msg.sender, _to, _value);
83    return true;
84  }

```

✓ The code meets the specification.

## Formal Verification Request 32

If method completes, integer overflow would not happen.



31, Jul 2019



165.57 ms

Line 94 in File ERC20Token.sol

```

94  //@CTK_NO_OVERFLOW

```

Line 107-111 in File ERC20Token.sol

```

107 function transferFrom(address _from, address _to, uint256 _value) public returns (
    bool success) {
108   _transfer(_from, _to, _value);
109   _approve(_from, msg.sender, _allowed[_from][msg.sender].sub(_value));
110   return true;
111 }

```

✓ The code meets the specification.

## Formal Verification Request 33

Buffer overflow / array index out of bound would never happen.



31, Jul 2019



21.16 ms

Line 95 in File ERC20Token.sol

```

95  //@CTK_NO_BUF_OVERFLOW

```

Line 107-111 in File ERC20Token.sol

```

107 function transferFrom(address _from, address _to, uint256 _value) public returns (
    bool success) {
108   _transfer(_from, _to, _value);
109   _approve(_from, msg.sender, _allowed[_from][msg.sender].sub(_value));
110   return true;
111 }

```

✓ The code meets the specification.

## Formal Verification Request 34

Method will not encounter an assertion failure.

📅 31, Jul 2019

🕒 22.03 ms

Line 96 in File ERC20Token.sol

96 `//@CTK NO_ASF`

Line 107-111 in File ERC20Token.sol

```

107  function transferFrom(address _from, address _to, uint256 _value) public returns (
      bool success) {
108      _transfer(_from, _to, _value);
109      _approve(_from, msg.sender, _allowed[_from][msg.sender].sub(_value));
110      return true;
111  }

```

✅ The code meets the specification.

## Formal Verification Request 35

transferFrom correctness

📅 31, Jul 2019

🕒 450.99 ms

Line 97-106 in File ERC20Token.sol

```

97  /*@CTK "transferFrom correctness"
98      @tag assume_completion
99      @post _to != 0x0
100      @post _value <= _balances[_from] && _value <= _allowed[_from][msg.sender]
101      @post _to != _from -> __post._balances[_from] == _balances[_from] - _value
102      @post _to != _from -> __post._balances[_to] == _balances[_to] + _value
103      @post _to == _from -> __post._balances[_from] == _balances[_from]
104      @post __post._allowed[_from][msg.sender] == _allowed[_from][msg.sender] - _value
105      @post success == true
106  */

```

Line 107-111 in File ERC20Token.sol

```

107  function transferFrom(address _from, address _to, uint256 _value) public returns (
      bool success) {
108      _transfer(_from, _to, _value);
109      _approve(_from, msg.sender, _allowed[_from][msg.sender].sub(_value));
110      return true;
111  }

```

✅ The code meets the specification.

## Formal Verification Request 36

If method completes, integer overflow would not happen.

📅 31, Jul 2019

🕒 34.44 ms

Line 122 in File ERC20Token.sol

```
122  // @CTK_NO_OVERFLOW
```

Line 129-132 in File ERC20Token.sol

```
129  function approve(address _spender, uint256 _value) public returns (bool success) {  
130      _approve(msg.sender, _spender, _value);  
131      return true;  
132  }
```

✓ The code meets the specification.

## Formal Verification Request 37

Buffer overflow / array index out of bound would never happen.

📅 31, Jul 2019

🕒 0.61 ms

Line 123 in File ERC20Token.sol

```
123  // @CTK_NO_BUF_OVERFLOW
```

Line 129-132 in File ERC20Token.sol

```
129  function approve(address _spender, uint256 _value) public returns (bool success) {  
130      _approve(msg.sender, _spender, _value);  
131      return true;  
132  }
```

✓ The code meets the specification.

## Formal Verification Request 38

Method will not encounter an assertion failure.

📅 31, Jul 2019

🕒 0.59 ms

Line 124 in File ERC20Token.sol

```
124  // @CTK_NO_ASF
```

Line 129-132 in File ERC20Token.sol

```
129  function approve(address _spender, uint256 _value) public returns (bool success) {  
130      _approve(msg.sender, _spender, _value);  
131      return true;  
132  }
```

✓ The code meets the specification.

## Formal Verification Request 39

approve correctness

📅 31, Jul 2019

🕒 6.92 ms

Line 125-128 in File ERC20Token.sol

```
125  /*@CTK "approve correctness"
126     @tag assume_completion
127     @post __post._allowed[msg.sender][_spender] == _value
128  */
```

Line 129-132 in File ERC20Token.sol

```
129  function approve(address _spender, uint256 _value) public returns (bool success) {
130      _approve(msg.sender, _spender, _value);
131      return true;
132  }
```

✅ The code meets the specification.

## Formal Verification Request 40

If method completes, integer overflow would not happen.

📅 31, Jul 2019

🕒 62.55 ms

Line 144 in File ERC20Token.sol

```
144  //@CTK NO_OVERFLOW
```

Line 151-154 in File ERC20Token.sol

```
151  function increaseAllowance(address _spender, uint256 _addedValue) public returns (
152      bool) {
153      _approve(msg.sender, _spender, _allowed[msg.sender][_spender].add(_addedValue));
154      return true;
155  }
```

✅ The code meets the specification.

## Formal Verification Request 41

Buffer overflow / array index out of bound would never happen.

📅 31, Jul 2019

🕒 0.91 ms

Line 145 in File ERC20Token.sol

```
145  //@CTK NO_BUF_OVERFLOW
```

Line 151-154 in File ERC20Token.sol

```

151 function increaseAllowance(address _spender, uint256 _addedValue) public returns (
    bool) {
152     _approve(msg.sender, _spender, _allowed[msg.sender][_spender].add(_addedValue));
153     return true;
154 }


```

✓ The code meets the specification.

## Formal Verification Request 42

Method will not encounter an assertion failure.

 31, Jul 2019

 0.86 ms

Line 146 in File ERC20Token.sol

```

146 // @CTK NO_ASF

```

Line 151-154 in File ERC20Token.sol

```

151 function increaseAllowance(address _spender, uint256 _addedValue) public returns (
    bool) {
152     _approve(msg.sender, _spender, _allowed[msg.sender][_spender].add(_addedValue));
153     return true;
154 }


```

✓ The code meets the specification.

## Formal Verification Request 43

increaseApproval correctness

 31, Jul 2019

 7.5 ms

Line 147-150 in File ERC20Token.sol

```

147 /*@CTK "increaseApproval correctness"
148    @tag assume_completion
149    @post __post._allowed[msg.sender][_spender] == _allowed[msg.sender][_spender] +
        _addedValue
150 */

```

Line 151-154 in File ERC20Token.sol

```

151 function increaseAllowance(address _spender, uint256 _addedValue) public returns (
    bool) {
152     _approve(msg.sender, _spender, _allowed[msg.sender][_spender].add(_addedValue));
153     return true;
154 }

```

✓ The code meets the specification.

## Formal Verification Request 44

If method completes, integer overflow would not happen.

📅 31, Jul 2019

🕒 52.33 ms

Line 166 in File ERC20Token.sol

166 `//@CTK NO_OVERFLOW`

Line 180-183 in File ERC20Token.sol

```
180 function decreaseAllowance(address _spender, uint256 _subtractedValue) public
    returns (bool) {
181     _approve(msg.sender, _spender, _allowed[msg.sender][_spender].sub(_subtractedValue
        ));
182     return true;
183 }
```

✅ The code meets the specification.

## Formal Verification Request 45

Buffer overflow / array index out of bound would never happen.

📅 31, Jul 2019

🕒 0.99 ms

Line 167 in File ERC20Token.sol

167 `//@CTK NO_BUF_OVERFLOW`

Line 180-183 in File ERC20Token.sol

```
180 function decreaseAllowance(address _spender, uint256 _subtractedValue) public
    returns (bool) {
181     _approve(msg.sender, _spender, _allowed[msg.sender][_spender].sub(_subtractedValue
        ));
182     return true;
183 }
```

✅ The code meets the specification.

## Formal Verification Request 46

Method will not encounter an assertion failure.

📅 31, Jul 2019

🕒 0.92 ms

Line 168 in File ERC20Token.sol

168 `//@CTK NO_ASF`

Line 180-183 in File ERC20Token.sol

```

180 function decreaseAllowance(address _spender, uint256 _subtractedValue) public
      returns (bool) {
181     _approve(msg.sender, _spender, _allowed[msg.sender][_spender].sub(_subtractedValue
          ));
182     return true;
183 }

```

✓ The code meets the specification.

## Formal Verification Request 47

### decreaseApproval0



31, Jul 2019



19.62 ms

Line 169-173 in File ERC20Token.sol

```

169 /*@CTK decreaseApproval0
170     @pre __return == true
171     @pre _allowed[msg.sender][_spender] <= _subtractedValue
172     @post __post._allowed[msg.sender][_spender] == 0
173 */

```

Line 180-183 in File ERC20Token.sol

```

180 function decreaseAllowance(address _spender, uint256 _subtractedValue) public
      returns (bool) {
181     _approve(msg.sender, _spender, _allowed[msg.sender][_spender].sub(_subtractedValue
          ));
182     return true;
183 }

```

✓ The code meets the specification.

## Formal Verification Request 48

### decreaseApproval



31, Jul 2019



7.52 ms

Line 174-179 in File ERC20Token.sol

```

174 /*@CTK decreaseApproval
175     @pre __return == true
176     @pre _allowed[msg.sender][_spender] > _subtractedValue
177     @post __post._allowed[msg.sender][_spender] ==
178         _allowed[msg.sender][_spender] - _subtractedValue
179 */

```

Line 180-183 in File ERC20Token.sol

```

180 function decreaseAllowance(address _spender, uint256 _subtractedValue) public
      returns (bool) {
181     _approve(msg.sender, _spender, _allowed[msg.sender][_spender].sub(_subtractedValue
          ));

```



```
182     return true;
183 }
```

✓ The code meets the specification.

## Formal Verification Request 49

If method completes, integer overflow would not happen.

31, Jul 2019

206.98 ms

Line 28 in File MainToken.sol

```
28  //@CTK NO_OVERFLOW
```

Line 39-51 in File MainToken.sol

```
39  function mint(uint256 _amount) onlyOwner() external {
40      require(!mintingStatus);
41      uint newTotalSupply = totalSupply.add(_amount);
42      address tokenOwner = owner();
43
44      require( newTotalSupply <= maxSupply );
45
46      _balances[tokenOwner] = _balances[tokenOwner].add(_amount);
47
48      totalSupply = newTotalSupply;
49
50      emit Minted(tokenOwner, _amount, _amount);
51 }
```

✓ The code meets the specification.

## Formal Verification Request 50

Buffer overflow / array index out of bound would never happen.

31, Jul 2019

27.15 ms

Line 29 in File MainToken.sol

```
29  //@CTK NO_BUF_OVERFLOW
```

Line 39-51 in File MainToken.sol

```
39  function mint(uint256 _amount) onlyOwner() external {
40      require(!mintingStatus);
41      uint newTotalSupply = totalSupply.add(_amount);
42      address tokenOwner = owner();
43
44      require( newTotalSupply <= maxSupply );
45
46      _balances[tokenOwner] = _balances[tokenOwner].add(_amount);
47
48      totalSupply = newTotalSupply;
```

```

49
50     emit Minted(tokenOwner, _amount, _amount);
51 }


```

✓ The code meets the specification.

## Formal Verification Request 51

Method will not encounter an assertion failure.

 31, Jul 2019

 26.23 ms

Line 30 in File MainToken.sol

```

30 // @CTK NO_ASF

```

Line 39-51 in File MainToken.sol

```

39 function mint(uint256 _amount) onlyOwner() external {
40     require(!mintingStatus);
41     uint newTotalSupply = totalSupply.add(_amount);
42     address tokenOwner = owner();
43
44     require( newTotalSupply <= maxSupply );
45
46     _balances[tokenOwner] = _balances[tokenOwner].add(_amount);
47
48     totalSupply = newTotalSupply;
49
50     emit Minted(tokenOwner, _amount, _amount);
51 }


```

✓ The code meets the specification.

## Formal Verification Request 52

mint

 31, Jul 2019

 275.43 ms

Line 31-38 in File MainToken.sol

```

31 /*@CTK "mint"
32     @tag assume_completion
33     @post mintingStatus == false
34     @post _owner == msg.sender
35     @post __post.totalSupply == totalSupply + _amount
36     @post __post.totalSupply <= maxSupply
37     @post __post._balances[msg.sender] == _balances[msg.sender] + _amount
38 */

```

Line 39-51 in File MainToken.sol

```
39 function mint(uint256 _amount) onlyOwner() external {
40     require(!mintingStatus);
41     uint newTotalSupply = totalSupply.add(_amount);
42     address tokenOwner = owner();
43
44     require( newTotalSupply <= maxSupply );
45
46     _balances[tokenOwner] = _balances[tokenOwner].add(_amount);
47
48     totalSupply = newTotalSupply;
49
50     emit Minted(tokenOwner, _amount, _amount);
51 }
```

✓ The code meets the specification.

## Formal Verification Request 53

lockAccount correctness

📅 31, Jul 2019

🕒 32.22 ms

Line 53-58 in File MainToken.sol

```
53 /*@CTK "lockAccount correctness"
54     @tag assume_completion
55     @post isLocked[_account] == false
56     @post _owner == msg.sender
57     @post __post.isLocked[_account] == true
58 */
```

Line 59-63 in File MainToken.sol

```
59 function lockAccount(address _account) onlyOwner() external {
60     require(!isLocked[_account]);
61
62     isLocked[_account] = true;
63 }
```

✓ The code meets the specification.

## Formal Verification Request 54

unlockAccount correctness

📅 31, Jul 2019

🕒 31.55 ms

Line 65-70 in File MainToken.sol

```
65 /*@CTK "unlockAccount correctness"
66     @tag assume_completion
67     @post isLocked[_account] == true
68     @post _owner == msg.sender
69     @post __post.isLocked[_account] == false
70 */
```

Line 71-75 in File MainToken.sol

```
71  function unlockAccount(address _account) onlyOwner() external {
72      require(isLocked[_account]);
73
74      isLocked[_account] = false;
75  }
```

✓ The code meets the specification.

## Formal Verification Request 55

paused correctness



31, Jul 2019



4.42 ms

Line 80-83 in File MainToken.sol

```
80  /*@CTK "paused correctness"
81      @tag assume_completion
82      @post __return == _paused
83  */
```

Line 84-86 in File MainToken.sol

```
84  function paused() public view returns (bool) {
85      return _paused;
86  }
```

✓ The code meets the specification.

## Formal Verification Request 56

pause correctness



31, Jul 2019



37.96 ms

Line 107-112 in File MainToken.sol

```
107 /*@CTK "pause correctness"
108     @tag assume_completion
109     @post _paused == false
110     @post _owner == msg.sender
111     @post __post._paused == true
112 */
```

Line 113-116 in File MainToken.sol

```
113 function pause() public onlyOwner whenNotPaused {
114     _paused = true;
115     emit Paused(msg.sender);
116 }
```

✓ The code meets the specification.

## Formal Verification Request 57

unpause correctness

📅 31, Jul 2019

🕒 35.62 ms

Line 121-126 in File MainToken.sol

```
121  /*@CTK "unpause correctness"
122     @tag assume_completion
123     @post _paused == true
124     @post _owner == msg.sender
125     @post __post._paused == false
126  */
```

Line 127-130 in File MainToken.sol

```
127  function unpause() public onlyOwner whenPaused {
128      _paused = false;
129      emit Unpaused(msg.sender);
130  }
```

✅ The code meets the specification.

## Formal Verification Request 58

If method completes, integer overflow would not happen.

📅 31, Jul 2019

🕒 274.03 ms

Line 132 in File MainToken.sol

```
132  //@CTK NO_OVERFLOW
```

Line 146-148 in File MainToken.sol

```
146  function transfer(address _to, uint256 _value) public onlyUnlocked whenNotPaused
      returns (bool) {
147      return super.transfer(_to, _value);
148  }
```

✅ The code meets the specification.

## Formal Verification Request 59

Buffer overflow / array index out of bound would never happen.

📅 31, Jul 2019

🕒 8.76 ms

Line 133 in File MainToken.sol

```
133  //@CTK NO_BUF_OVERFLOW
```

Line 146-148 in File MainToken.sol

```

146 function transfer(address _to, uint256 _value) public onlyUnlocked whenNotPaused
    returns (bool) {
147     return super.transfer(_to, _value);
148 }

```

✓ The code meets the specification.

## Formal Verification Request 60

Method will not encounter an assertion failure.

📅 31, Jul 2019

🕒 10.02 ms

Line 134 in File MainToken.sol

```

134 // @CTK NO_ASF

```

Line 146-148 in File MainToken.sol

```

146 function transfer(address _to, uint256 _value) public onlyUnlocked whenNotPaused
    returns (bool) {
147     return super.transfer(_to, _value);
148 }

```

✓ The code meets the specification.

## Formal Verification Request 61

transfer correctness

📅 31, Jul 2019

🕒 275.36 ms

Line 135-145 in File MainToken.sol

```

135 /*@CTK "transfer correctness"
136   @tag assume_completion
137   @post _paused == false
138   @post isLocked[msg.sender] == false
139   @post _to != 0x0
140   @post _value <= _balances[msg.sender]
141   @post _to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender] -
    _value
142   @post _to != msg.sender -> __post._balances[_to] == _balances[_to] + _value
143   @post _to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
144   @post __return == true
145 */

```

Line 146-148 in File MainToken.sol

```

146 function transfer(address _to, uint256 _value) public onlyUnlocked whenNotPaused
    returns (bool) {
147     return super.transfer(_to, _value);
148 }


```

✓ The code meets the specification.

## Formal Verification Request 62

If method completes, integer overflow would not happen.

 31, Jul 2019

 333.82 ms

Line 150 in File MainToken.sol

150 `//@CTK_NO_OVERFLOW`

Line 165-169 in File MainToken.sol


```
165 function transferFrom(address _from, address _to, uint256 _value) public
    onlyUnlocked
166 whenNotPaused returns
167 (bool) {
168     return super.transferFrom(_from, _to, _value);
169 }
```

 The code meets the specification.

## Formal Verification Request 63

Buffer overflow / array index out of bound would never happen.

 31, Jul 2019

 41.78 ms

Line 151 in File MainToken.sol

151 `//@CTK_NO_BUF_OVERFLOW`

Line 165-169 in File MainToken.sol


```
165 function transferFrom(address _from, address _to, uint256 _value) public
    onlyUnlocked
166 whenNotPaused returns
167 (bool) {
168     return super.transferFrom(_from, _to, _value);
169 }
```

 The code meets the specification.

## Formal Verification Request 64

Method will not encounter an assertion failure.

 31, Jul 2019

 44.79 ms

Line 152 in File MainToken.sol

152 `//@CTK_NO_ASF`

Line 165-169 in File MainToken.sol

```

165 function transferFrom(address _from, address _to, uint256 _value) public
    onlyUnlocked
166 whenNotPaused returns
167 (bool) {
168     return super.transferFrom(_from, _to, _value);
169 }

```

✓ The code meets the specification.

## Formal Verification Request 65

transferFrom correctness

📅 31, Jul 2019

🕒 614.57 ms

Line 153-164 in File MainToken.sol

```

153 /*@CTK "transferFrom correctness"
154     @tag assume_completion
155     @post _to != 0x0
156     @post _paused == false
157     @post isLocked[msg.sender] == false
158     @post _value <= _balances[_from] && _value <= _allowed[_from][msg.sender]
159     @post _to != _from -> __post._balances[_from] == _balances[_from] - _value
160     @post _to != _from -> __post._balances[_to] == _balances[_to] + _value
161     @post _to == _from -> __post._balances[_from] == _balances[_from]
162     @post __post._allowed[_from][msg.sender] == _allowed[_from][msg.sender] - _value
163     @post __return == true
164 */

```

Line 165-169 in File MainToken.sol

```

165 function transferFrom(address _from, address _to, uint256 _value) public
    onlyUnlocked
166 whenNotPaused returns
167 (bool) {
168     return super.transferFrom(_from, _to, _value);
169 }

```

✓ The code meets the specification.

## Formal Verification Request 66

If method completes, integer overflow would not happen.

📅 31, Jul 2019

🕒 98.66 ms

Line 171 in File MainToken.sol

```

171 // @CTK NO_OVERFLOW

```

Line 180-183 in File MainToken.sol



```
180 function approve(address _spender, uint256 _value) public onlyUnlocked whenNotPaused
    returns
181 (bool) {
182     return super.approve(_spender, _value);
183 }
```

✓ The code meets the specification.

## Formal Verification Request 67

Buffer overflow / array index out of bound would never happen.

📅 31, Jul 2019

🕒 5.07 ms

Line 172 in File MainToken.sol

```
172 //@CTK_NO_BUF_OVERFLOW
```

Line 180-183 in File MainToken.sol

```
180 function approve(address _spender, uint256 _value) public onlyUnlocked whenNotPaused
    returns
181 (bool) {
182     return super.approve(_spender, _value);
183 }
```

✓ The code meets the specification.

## Formal Verification Request 68

Method will not encounter an assertion failure.

📅 31, Jul 2019

🕒 5.24 ms

Line 173 in File MainToken.sol

```
173 //@CTK_NO_ASF
```

Line 180-183 in File MainToken.sol

```
180 function approve(address _spender, uint256 _value) public onlyUnlocked whenNotPaused
    returns
181 (bool) {
182     return super.approve(_spender, _value);
183 }
```

✓ The code meets the specification.

## Formal Verification Request 69

approve correctness

📅 31, Jul 2019

🕒 10.47 ms

Line 174-179 in File MainToken.sol

```
174  /*@CTK "approve correctness"
175     @post _paused == false
176     @post isLocked[msg.sender] == false
177     @tag assume_completion
178     @post __post._allowed[msg.sender][_spender] == _value
179  */
```

Line 180-183 in File MainToken.sol

```
180  function approve(address _spender, uint256 _value) public onlyUnlocked whenNotPaused
      returns
181  (bool) {
182      return super.approve(_spender, _value);
183  }
```

✓ The code meets the specification.

## Formal Verification Request 70

If method completes, integer overflow would not happen.



31, Jul 2019



146.82 ms

Line 185 in File MainToken.sol

```
185  //@CTK NO_OVERFLOW
```

Line 194-197 in File MainToken.sol

```
194  function increaseAllowance(address _spender, uint _addedValue) public onlyUnlocked
      whenNotPaused
195  returns (bool) {
196      return super.increaseAllowance(_spender, _addedValue);
197  }
```

✓ The code meets the specification.

## Formal Verification Request 71

Buffer overflow / array index out of bound would never happen.



31, Jul 2019



6.57 ms

Line 186 in File MainToken.sol

```
186  //@CTK NO_BUF_OVERFLOW
```

Line 194-197 in File MainToken.sol

```
194  function increaseAllowance(address _spender, uint _addedValue) public onlyUnlocked
      whenNotPaused
195  returns (bool) {
196      return super.increaseAllowance(_spender, _addedValue);
197  }
```

✓ The code meets the specification.

## Formal Verification Request 72

Method will not encounter an assertion failure.

📅 31, Jul 2019

🕒 6.47 ms

Line 187 in File MainToken.sol

```
187 // @CTK NO_ASF
```

Line 194-197 in File MainToken.sol

```
194 function increaseAllowance(address _spender, uint _addedValue) public onlyUnlocked
    whenNotPaused
195 returns (bool) {
196     return super.increaseAllowance(_spender, _addedValue);
197 }
```

✓ The code meets the specification.

## Formal Verification Request 73

increaseApproval correctness

📅 31, Jul 2019

🕒 60.81 ms

Line 188-193 in File MainToken.sol

```
188 /* @CTK "increaseApproval correctness"
189     @tag assume_completion
190     @post _paused == false
191     @post isLocked[msg.sender] == false
192     @post __post._allowed[msg.sender][_spender] == _allowed[msg.sender][_spender] +
        _addedValue
193 */
```

Line 194-197 in File MainToken.sol

```
194 function increaseAllowance(address _spender, uint _addedValue) public onlyUnlocked
    whenNotPaused
195 returns (bool) {
196     return super.increaseAllowance(_spender, _addedValue);
197 }
```

✓ The code meets the specification.

## Formal Verification Request 74

If method completes, integer overflow would not happen.

📅 31, Jul 2019

🕒 129.68 ms

Line 199 in File MainToken.sol

```
199 // @CTK_NO_OVERFLOW
```

Line 217-221 in File MainToken.sol

```
217 function decreaseAllowance(address _spender, uint _subtractedValue) public  
    onlyUnlocked  
218 whenNotPaused returns  
219 (bool) {  
220     return super.decreaseAllowance(_spender, _subtractedValue);  
221 }
```

✓ The code meets the specification.

## Formal Verification Request 75

Buffer overflow / array index out of bound would never happen.

📅 31, Jul 2019

🕒 6.88 ms

Line 200 in File MainToken.sol

```
200 // @CTK_NO_BUF_OVERFLOW
```

Line 217-221 in File MainToken.sol

```
217 function decreaseAllowance(address _spender, uint _subtractedValue) public  
    onlyUnlocked  
218 whenNotPaused returns  
219 (bool) {  
220     return super.decreaseAllowance(_spender, _subtractedValue);  
221 }
```

✓ The code meets the specification.

## Formal Verification Request 76

Method will not encounter an assertion failure.

📅 31, Jul 2019

🕒 5.85 ms

Line 201 in File MainToken.sol

```
201 // @CTK_NO_ASF
```

Line 217-221 in File MainToken.sol

```
217 function decreaseAllowance(address _spender, uint _subtractedValue) public  
    onlyUnlocked  
218 whenNotPaused returns  
219 (bool) {  
220     return super.decreaseAllowance(_spender, _subtractedValue);  
221 }
```

✓ The code meets the specification.

## Formal Verification Request 77

### decreaseApproval0

📅 31, Jul 2019

🕒 32.6 ms

Line 202-208 in File MainToken.sol

```
202  /*@CTK decreaseApproval0
203     @pre __return == true
204     @pre _allowed[msg.sender][_spender] <= _subtractedValue
205     @post _paused == false
206     @post isLocked[msg.sender] == false
207     @post __post._allowed[msg.sender][_spender] == 0
208  */
```

Line 217-221 in File MainToken.sol

```
217  function decreaseAllowance(address _spender, uint _subtractedValue) public
      onlyUnlocked
218  whenNotPaused returns
219  (bool) {
220      return super.decreaseAllowance(_spender, _subtractedValue);
221  }
```

✅ The code meets the specification.

## Formal Verification Request 78

### decreaseApproval

📅 31, Jul 2019

🕒 36.53 ms

Line 209-216 in File MainToken.sol

```
209  /*@CTK decreaseApproval
210     @pre __return == true
211     @pre _allowed[msg.sender][_spender] > _subtractedValue
212     @post _paused == false
213     @post isLocked[msg.sender] == false
214     @post __post._allowed[msg.sender][_spender] ==
215           _allowed[msg.sender][_spender] - _subtractedValue
216  */
```

Line 217-221 in File MainToken.sol

```
217  function decreaseAllowance(address _spender, uint _subtractedValue) public
      onlyUnlocked
218  whenNotPaused returns
219  (bool) {
220      return super.decreaseAllowance(_spender, _subtractedValue);
221  }
```

✅ The code meets the specification.

## Formal Verification Request 79

SafeMath mul

📅 31, Jul 2019

🕒 316.77 ms

Line 11-17 in File SafeMath.sol

```
11  /*@CTK "SafeMath mul"
12     @post (((a) > (0)) && (((a) * (b)) / (a)) != (b))) == (__reverted)
13     @post !__reverted -> __return == a * b
14     @post !__reverted == !__has_overflow
15     @post !(__has_buf_overflow)
16     @post !(__has_assertion_failure)
17 */
```

Line 18-30 in File SafeMath.sol

```
18  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
19      // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
20      // benefit is lost if 'b' is also tested.
21      // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
22      if (a == 0) {
23          return 0;
24      }
25
26      uint256 c = a * b;
27      require(c / a == b);
28
29      return c;
30 }
```

✅ The code meets the specification.

## Formal Verification Request 80

SafeMath div

📅 31, Jul 2019

🕒 12.44 ms

Line 35-41 in File SafeMath.sol

```
35  /*@CTK "SafeMath div"
36     @post b != 0 -> !__reverted
37     @post !__reverted -> __return == a / b
38     @post !__reverted -> !__has_overflow
39     @post !(__has_buf_overflow)
40     @post !(__has_assertion_failure)
41 */
```

Line 42-49 in File SafeMath.sol

```
42  function div(uint256 a, uint256 b) internal pure returns (uint256) {
43      // Solidity only automatically asserts when dividing by 0
44      require(b > 0);
45      uint256 c = a / b;
46      // assert(a == b * c + a % b); // There is no case in which this doesn't hold
```


```
47
48     return c;
49 }
```

✓ The code meets the specification.

## Formal Verification Request 81

SafeMath sub

 31, Jul 2019

 10.91 ms

Line 54-60 in File SafeMath.sol

```
54  /*@CTK "SafeMath sub"
55     @post (a < b) == __reverted
56     @post !__reverted -> __return == a - b
57     @post !__reverted -> !__has_overflow
58     @post !(__has_buf_overflow)
59     @post !(__has_assertion_failure)
60 */
```

Line 61-66 in File SafeMath.sol


```
61  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
62     require(b <= a);
63     uint256 c = a - b;
64
65     return c;
66 }
```

✓ The code meets the specification.

## Formal Verification Request 82

SafeMath add

 31, Jul 2019

 13.86 ms

Line 71-77 in File SafeMath.sol

```
71  /*@CTK "SafeMath add"
72     @post (a + b < a || a + b < b) == __reverted
73     @post !__reverted -> __return == a + b
74     @post !__reverted -> !__has_overflow
75     @post !(__has_buf_overflow)
76     @post !(__has_assertion_failure)
77 */
```

Line 78-83 in File SafeMath.sol

```
78  function add(uint256 a, uint256 b) internal pure returns (uint256) {
79     uint256 c = a + b;
80     require(c >= a);
81 }
```

```
82     return c;  
83 }
```

✓ The code meets the specification.

## Formal Verification Request 83

SafeMath div

📅 31, Jul 2019

🕒 10.43 ms

Line 89-95 in File SafeMath.sol

```
89  /*@CTK "SafeMath div"  
90     @post b != 0 -> !__reverted  
91     @post !__reverted -> __return == a % b  
92     @post !__reverted -> !__has_overflow  
93     @post !(__has_buf_overflow)  
94     @post !(__has_assertion_failure)  
95  */
```

Line 96-99 in File SafeMath.sol

```
96  function mod(uint256 a, uint256 b) internal pure returns (uint256) {  
97      require(b != 0);  
98      return a % b;  
99  }
```

✓ The code meets the specification.



## Source Code with CertiK Labels

File LinearMintableToken.sol

```

1  pragma solidity ^0.4.24;
2
3  import "./ERC20Token.sol";
4  import "./Ownable.sol";
5
6  contract LinearMintableToken is ERC20Token, Ownable {
7      event Minted(address recipient, uint256 mintingAmount, uint256 mintedAmount);
8
9      uint256 constant SECONDS_IN_A_DAY = 86400;
10
11     bool public mintingStatus = false;
12
13     uint256 public mintingSupply;
14     uint256 public intervalPeriodInDays;
15     uint256 public intervalCount;
16     uint256 public mintAmountPerPeriod;
17     uint256 public createdTimestamp;
18     uint256 public mintedAmount;
19
20     //@CTK NO_OVERFLOW
21     //@CTK NO_BUF_OVERFLOW
22     //@CTK NO_ASF
23     /*@CTK "registerLinearMint correctness"
24         @tag assume_completion
25         @post mintingStatus == false
26         @post totalSupply + _mintingSupply <= maxSupply
27         @post _owner == msg.sender
28         @post __post.mintingStatus == true
29         @post __post.mintingSupply == _mintingSupply
30         @post __post.mintingSupply > 0
31         @post __post.mintingSupply == _mintingSupply
32         @post __post.mintAmountPerPeriod == _mintAmountPerPeriod
33         @post __post.mintAmountPerPeriod > 0
34         @post __post.intervalPeriodInDays == _intervalPeriodInDays
35         @post __post.intervalPeriodInDays > 0
36         @post __post.createdTimestamp == block.timestamp
37     */
38     function registerLinearMint(
39         uint256 _mintingSupply,
40         uint256 _mintAmountPerPeriod,
41         uint256 _intervalPeriodInDays
42     ) external onlyOwner() {
43         require(!mintingStatus);
44         require(_mintingSupply > 0);
45         require(totalSupply.add(_mintingSupply) <= maxSupply);
46         require(_mintAmountPerPeriod > 0);
47         require(_intervalPeriodInDays > 0);
48
49         mintingStatus = true;
50
51         mintingSupply = _mintingSupply;
52         mintAmountPerPeriod = _mintAmountPerPeriod;
53         intervalPeriodInDays = _intervalPeriodInDays;
54         createdTimestamp = block.timestamp;

```

```

55 }
56
57 function linearMint() external {
58     mintInternal(block.timestamp);
59 }
60
61 //@CTK FAIL NO_OVERFLOW
62 //@CTK NO_BUF_OVERFLOW
63 /*CTK "mintInternal NO_ASF and correctness"
64     @tag assume_completion
65     @pre intervalPeriodInDays > 0
66     @pre SECONDS_IN_A_DAY > 0
67     @post mintingStatus == true
68     @post !(__has_assertion_failure)
69 */
70 function mintInternal(uint256 blockTimestamp) internal {
71     require(mintingStatus);
72
73     address tokenOwner = owner();
74     uint256 mintingAmount = calculateMintAmount(blockTimestamp);
75     mintingAmount = mintingAmount.sub(mintedAmount);
76
77     if ( mintingAmount == 0 )
78         return;
79
80     if ( mintingAmount >= mintingSupply ) {
81         mintingAmount = mintingSupply.sub(mintedAmount);
82         mintingStatus = false;
83     }
84
85     totalSupply = totalSupply.add(mintingAmount);
86
87     _balances[tokenOwner] = _balances[tokenOwner].add(mintingAmount);
88
89     mintedAmount = mintedAmount.add(mintingAmount);
90
91     emit Minted(tokenOwner, mintingAmount, mintedAmount);
92 }
93
94 //@CTK FAIL NO_OVERFLOW
95 //@CTK NO_BUF_OVERFLOW
96 /*CTK "calculateMintAmount NO_ASF"
97     @tag assume_completion
98     @pre intervalPeriodInDays > 0
99     @pre SECONDS_IN_A_DAY > 0
100     @post !(__has_assertion_failure)
101 */
102 function calculateMintAmount(uint256 blockTimestamp) public view returns (uint256
    mintAmount) {
103     uint256 pastDays = blockTimestamp.sub(createdTimestamp).div(SECONDS_IN_A_DAY);
104     uint256 pastIntervals = pastDays / intervalPeriodInDays + 1;
105
106     return pastIntervals * mintAmountPerPeriod;
107 }
108 }

```

File Ownable.sol

```
1 pragma solidity ^0.4.24;
```

```

2
3 /**
4  * @title Ownable
5  * @dev The Ownable contract has an owner address, and provides basic authorization
        control
6  * functions, this simplifies the implementation of "user permissions".
7  */
8 contract Ownable {
9     address private _owner;
10
11     event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);
12
13     /**
14      * @dev The Ownable constructor sets the original 'owner' of the contract to the
        sender
15      * account.
16      */
17     /*@CTK Ownable
18      @post __post._owner == msg.sender
19      */
20     constructor () internal {
21         _owner = msg.sender;
22         emit OwnershipTransferred(address(0), _owner);
23     }
24
25     /**
26      * @return the address of the owner.
27      */
28     /*@CTK owner
29      @post __return == _owner
30      */
31     function owner() public view returns (address) {
32         return _owner;
33     }
34
35     /**
36      * @dev Throws if called by any account other than the owner.
37      */
38     modifier onlyOwner() {
39         require(isOwner());
40         _;
41     }
42
43     /**
44      * @return true if 'msg.sender' is the owner of the contract.
45      */
46     /*@CTK isOwner
47      @post __return == (msg.sender == _owner)
48      */
49     function isOwner() public view returns (bool) {
50         return msg.sender == _owner;
51     }
52
53     /**
54      * @dev Allows the current owner to relinquish control of the contract.
55      * It will not be possible to call the functions with the 'onlyOwner'
56      * modifier anymore.
57      * @notice Renouncing ownership will leave the contract without an owner,

```

```

58  * thereby removing any functionality that is only available to the owner.
59  */
60  // function renounceOwnership() public onlyOwner {
61  //     emit OwnershipTransferred(_owner, address(0));
62  //     _owner = address(0);
63  // }
64
65  /**
66   * @dev Allows the current owner to transfer control of the contract to a newOwner.
67   * @param newOwner The address to transfer ownership to.
68   */
69  /*@CTK transferOwnership
70   @tag assume_completion
71   @post _owner == msg.sender
72   */
73  function transferOwnership(address newOwner) public onlyOwner {
74      _transferOwnership(newOwner);
75  }
76
77  /**
78   * @dev Transfers control of the contract to a newOwner.
79   * @param newOwner The address to transfer ownership to.
80   */
81  /*@CTK _transferOwnership
82   @tag assume_completion
83   @post newOwner != address(0)
84   @post __post._owner == newOwner
85   */
86  function _transferOwnership(address newOwner) internal {
87      require(newOwner != address(0));
88      emit OwnershipTransferred(_owner, newOwner);
89      _owner = newOwner;
90  }
91 }

```

File ERC20Token.sol

```

1  pragma solidity ^0.4.24;
2
3  import "./SafeMath.sol";
4  import "./TokenRecipient.sol";
5
6  contract ERC20Token {
7      using SafeMath for uint;
8
9      string public name;
10     string public symbol;
11     uint256 public totalSupply;
12     uint256 public maxSupply;
13     uint8 public decimals;
14
15     mapping(address => uint256) _balances;
16
17     mapping (address => mapping (address => uint256)) private _allowed;
18
19     event Transfer(address from, address to, uint value);
20     event Approval(address from, address to, uint value);
21     event ApproveAndCall(address spender, uint value, bytes extraData);
22

```

```

23 //@CTK NO_OVERFLOW
24 //@CTK NO_BUF_OVERFLOW
25 //@CTK NO_ASF
26 /*@CTK "constructor correctness"
27   @tag assume_completion
28   @post __post._balances[msg.sender] == __post.totalSupply
29   @post __post._balances[msg.sender] == _initialSupply
30   @post __post.maxSupply == _maxSupply
31 */
32 constructor(string _name, string _symbol, uint8 _decimals, uint256 _initialSupply,
33   uint256 _maxSupply) public {
34   require(_maxSupply >= _initialSupply);
35   name = _name;
36   symbol = _symbol;
37   decimals = _decimals;
38
39   _balances[msg.sender] = _initialSupply;
40   totalSupply = _initialSupply;
41   maxSupply = _maxSupply;
42 }
43
44 //@CTK NO_OVERFLOW
45 //@CTK NO_BUF_OVERFLOW
46 //@CTK NO_ASF
47 /*@CTK "balanceOf correctness"
48   @post balance == _balances[_owner]
49 */
50 function balanceOf(address _owner) public view returns (uint balance) {
51   return _balances[_owner];
52 }
53
54 //@CTK NO_OVERFLOW
55 //@CTK NO_BUF_OVERFLOW
56 //@CTK NO_ASF
57 /*@CTK "allowance correctness"
58   @post remaining == _allowed[_owner][_spender]
59 */
60 function allowance(address _owner, address _spender) public view returns (uint256
61   remaining) {
62   return _allowed[_owner][_spender];
63 }
64 /**
65   * @dev Transfer token to a specified address.
66   * @param _to The address to transfer to.
67   * @param _value The amount to be transferred.
68   */
69 //@CTK NO_OVERFLOW
70 //@CTK NO_BUF_OVERFLOW
71 //@CTK NO_ASF
72 /*@CTK "transfer correctness"
73   @tag assume_completion
74   @post _to != 0x0
75   @post _value <= _balances[msg.sender]
76   @post _to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender] -
77     _value
78   @post _to != msg.sender -> __post._balances[_to] == _balances[_to] + _value

```

```

78     @post _to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
79     @post success == true
80     */
81     function transfer(address _to, uint256 _value) public returns (bool success) {
82         _transfer(msg.sender, _to, _value);
83         return true;
84     }
85
86     /**
87      * @dev Transfer tokens from one address to another.
88      * Note that while this function emits an Approval event, this is not required as
89      * per the specification,
90      * and other compliant implementations may not emit the event.
91      * @param _from address The address which you want to send tokens from
92      * @param _to address The address which you want to transfer to
93      * @param _value uint256 the amount of tokens to be transferred
94      */
95     // @CTK NO_OVERFLOW
96     // @CTK NO_BUF_OVERFLOW
97     // @CTK NO_ASF
98     /* @CTK "transferFrom correctness"
99      @tag assume_completion
100      @post _to != 0x0
101      @post _value <= _balances[_from] && _value <= _allowed[_from][msg.sender]
102      @post _to != _from -> __post._balances[_from] == _balances[_from] - _value
103      @post _to != _from -> __post._balances[_to] == _balances[_to] + _value
104      @post _to == _from -> __post._balances[_from] == _balances[_from]
105      @post __post._allowed[_from][msg.sender] == _allowed[_from][msg.sender] - _value
106      @post success == true
107      */
108     function transferFrom(address _from, address _to, uint256 _value) public returns (
109         bool success) {
110         _transfer(_from, _to, _value);
111         _approve(_from, msg.sender, _allowed[_from][msg.sender].sub(_value));
112         return true;
113     }
114
115     /**
116      * @dev Approve the passed address to spend the specified amount of tokens on
117      * behalf of msg.sender.
118      * Beware that changing an allowance with this method brings the risk that someone
119      * may use both the old
120      * and the new allowance by unfortunate transaction ordering. One possible
121      * solution to mitigate this
122      * race condition is to first reduce the spender's allowance to 0 and set the
123      * desired value afterwards:
124      * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
125      * @param _spender The address which will spend the funds.
126      * @param _value The amount of tokens to be spent.
127      */
128     // @CTK NO_OVERFLOW
129     // @CTK NO_BUF_OVERFLOW
130     // @CTK NO_ASF
131     /* @CTK "approve correctness"
132      @tag assume_completion
133      @post __post._allowed[msg.sender][_spender] == _value
134      */
135     function approve(address _spender, uint256 _value) public returns (bool success) {

```

```

130     _approve(msg.sender, _spender, _value);
131     return true;
132 }
133
134 /**
135  * @dev Increase the amount of tokens that an owner allowed to a spender.
136  * approve should be called when _allowed[msg.sender][spender] == 0. To increment
137  * allowed value is better to use this function to avoid 2 calls (and wait until
138  * the first transaction is mined)
139  * From MonolithDAO Token.sol
140  * Emits an Approval event.
141  * @param _spender The address which will spend the funds.
142  * @param _addedValue The amount of tokens to increase the allowance by.
143  */
144 // @CTK NO_OVERFLOW
145 // @CTK NO_BUF_OVERFLOW
146 // @CTK NO_ASF
147 /* @CTK "increaseApproval correctness"
148    @tag assume_completion
149    @post __post._allowed[msg.sender][_spender] == _allowed[msg.sender][_spender] +
150          _addedValue
151 */
152 function increaseAllowance(address _spender, uint256 _addedValue) public returns (
153     bool) {
154     _approve(msg.sender, _spender, _allowed[msg.sender][_spender].add(_addedValue));
155     return true;
156 }
157
158 /**
159  * @dev Decrease the amount of tokens that an owner allowed to a spender.
160  * approve should be called when _allowed[msg.sender][spender] == 0. To decrement
161  * allowed value is better to use this function to avoid 2 calls (and wait until
162  * the first transaction is mined)
163  * From MonolithDAO Token.sol
164  * Emits an Approval event.
165  * @param _spender The address which will spend the funds.
166  * @param _subtractedValue The amount of tokens to decrease the allowance by.
167  */
168 // @CTK NO_OVERFLOW
169 // @CTK NO_BUF_OVERFLOW
170 // @CTK NO_ASF
171 /* @CTK decreaseApproval0
172    @pre __return == true
173    @pre _allowed[msg.sender][_spender] <= _subtractedValue
174    @post __post._allowed[msg.sender][_spender] == 0
175 */
176 /* @CTK decreaseApproval
177    @pre __return == true
178    @pre _allowed[msg.sender][_spender] > _subtractedValue
179    @post __post._allowed[msg.sender][_spender] ==
180          _allowed[msg.sender][_spender] - _subtractedValue
181 */
182 function decreaseAllowance(address _spender, uint256 _subtractedValue) public
183     returns (bool) {
184     _approve(msg.sender, _spender, _allowed[msg.sender][_spender].sub(_subtractedValue));
185     return true;
186 }

```

```

184
185 function approveAndCall(address _spender, uint256 _value, bytes _extraData) public
    returns (bool success) {
186     require(_spender != address(0));
187
188     TokenRecipient spender = TokenRecipient(_spender);
189     if (approve(_spender, _value)) {
190         spender.receiveApproval(msg.sender, _value, this, _extraData);
191         emit ApproveAndCall(_spender, _value, _extraData );
192         return true;
193     }
194
195     return false;
196 }
197
198 /**
199  * @dev Approve an address to spend another addresses' tokens.
200  * @param owner The address that owns the tokens.
201  * @param spender The address that will spend the tokens.
202  * @param value The number of tokens that can be spent.
203  */
204 function _approve(address owner, address spender, uint256 value) internal {
205     require(spender != address(0));
206     require(owner != address(0));
207
208     _allowed[owner][spender] = value;
209     emit Approval(owner, spender, value);
210 }
211
212 /**
213  * @dev Transfer token for a specified addresses.
214  * @param from The address to transfer from.
215  * @param to The address to transfer to.
216  * @param value The amount to be transferred.
217  */
218 function _transfer(address from, address to, uint256 value) internal {
219     require(to != address(0));
220
221     _balances[from] = _balances[from].sub(value);
222     _balances[to] = _balances[to].add(value);
223     emit Transfer(from, to, value);
224 }
225 }

```

File MainToken.sol

```

1 pragma solidity ^0.4.24;
2
3 import "./LinearMintableToken.sol";
4
5 contract MainToken is LinearMintableToken {
6     event Paused(address account);
7     event Unpaused(address account);
8
9     mapping (address => bool) public isLocked;
10    bool private _paused;
11
12    constructor(
13        string _name,

```



```

14     string _symbol,
15     uint8 _decimals,
16     uint256 _initialSupply,
17     uint256 _maxSupply
18 ) ERC20Token(_name, _symbol, _decimals, _initialSupply, _maxSupply)
19 public {
20     _paused = false;
21 }
22
23 modifier onlyUnlocked() {
24     require(!isLocked[msg.sender], "msg.sender is locked");
25     _;
26 }
27
28 //@CTK NO_OVERFLOW
29 //@CTK NO_BUF_OVERFLOW
30 //@CTK NO_ASF
31 /*@CTK "mint"
32     @tag assume_completion
33     @post mintingStatus == false
34     @post _owner == msg.sender
35     @post __post.totalSupply == totalSupply + _amount
36     @post __post.totalSupply <= maxSupply
37     @post __post._balances[msg.sender] == _balances[msg.sender] + _amount
38 */
39 function mint(uint256 _amount) onlyOwner() external {
40     require(!mintingStatus);
41     uint newTotalSupply = totalSupply.add(_amount);
42     address tokenOwner = owner();
43
44     require( newTotalSupply <= maxSupply );
45
46     _balances[tokenOwner] = _balances[tokenOwner].add(_amount);
47
48     totalSupply = newTotalSupply;
49
50     emit Minted(tokenOwner, _amount, _amount);
51 }
52
53 /*@CTK "lockAccount correctness"
54     @tag assume_completion
55     @post isLocked[_account] == false
56     @post _owner == msg.sender
57     @post __post.isLocked[_account] == true
58 */
59 function lockAccount(address _account) onlyOwner() external {
60     require(!isLocked[_account]);
61
62     isLocked[_account] = true;
63 }
64
65 /*@CTK "unlockAccount correctness"
66     @tag assume_completion
67     @post isLocked[_account] == true
68     @post _owner == msg.sender
69     @post __post.isLocked[_account] == false
70 */
71 function unlockAccount(address _account) onlyOwner() external {

```

```

72     require(isLocked[_account]);
73
74     isLocked[_account] = false;
75 }
76
77 /**
78  * @return True if the contract is paused, false otherwise.
79  */
80 /*@CTK "paused correctness"
81  @tag assume_completion
82  @post __return == _paused
83  */
84 function paused() public view returns (bool) {
85     return _paused;
86 }
87
88 /**
89  * @dev Modifier to make a function callable only when the contract is not paused.
90  */
91 modifier whenNotPaused() {
92     require(!_paused, "Pausable: paused");
93     _;
94 }
95
96 /**
97  * @dev Modifier to make a function callable only when the contract is paused.
98  */
99 modifier whenPaused() {
100     require(_paused, "Pausable: not paused");
101     _;
102 }
103
104 /**
105  * @dev Called by a owner to pause, triggers stopped state.
106  */
107 /*@CTK "pause correctness"
108  @tag assume_completion
109  @post _paused == false
110  @post _owner == msg.sender
111  @post __post._paused == true
112  */
113 function pause() public onlyOwner whenNotPaused {
114     _paused = true;
115     emit Paused(msg.sender);
116 }
117
118 /**
119  * @dev Called by a owner to unpause, returns to normal state.
120  */
121 /*@CTK "unpause correctness"
122  @tag assume_completion
123  @post _paused == true
124  @post _owner == msg.sender
125  @post __post._paused == false
126  */
127 function unpause() public onlyOwner whenPaused {
128     _paused = false;
129     emit Unpaused(msg.sender);

```

```

130 }
131
132 //@CTK NO_OVERFLOW
133 //@CTK NO_BUF_OVERFLOW
134 //@CTK NO_ASF
135 /*@CTK "transfer correctness"
136   @tag assume_completion
137   @post _paused == false
138   @post isLocked[msg.sender] == false
139   @post _to != 0x0
140   @post _value <= _balances[msg.sender]
141   @post _to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender] -
     _value
142   @post _to != msg.sender -> __post._balances[_to] == _balances[_to] + _value
143   @post _to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
144   @post __return == true
145 */
146 function transfer(address _to, uint256 _value) public onlyUnlocked whenNotPaused
     returns (bool) {
147     return super.transfer(_to, _value);
148 }
149
150 //@CTK NO_OVERFLOW
151 //@CTK NO_BUF_OVERFLOW
152 //@CTK NO_ASF
153 /*@CTK "transferFrom correctness"
154   @tag assume_completion
155   @post _to != 0x0
156   @post _paused == false
157   @post isLocked[msg.sender] == false
158   @post _value <= _balances[_from] && _value <= _allowed[_from][msg.sender]
159   @post _to != _from -> __post._balances[_from] == _balances[_from] - _value
160   @post _to != _from -> __post._balances[_to] == _balances[_to] + _value
161   @post _to == _from -> __post._balances[_from] == _balances[_from]
162   @post __post._allowed[_from][msg.sender] == _allowed[_from][msg.sender] - _value
163   @post __return == true
164 */
165 function transferFrom(address _from, address _to, uint256 _value) public
     onlyUnlocked
166 whenNotPaused returns
167 (bool) {
168     return super.transferFrom(_from, _to, _value);
169 }
170
171 //@CTK NO_OVERFLOW
172 //@CTK NO_BUF_OVERFLOW
173 //@CTK NO_ASF
174 /*@CTK "approve correctness"
175   @post _paused == false
176   @post isLocked[msg.sender] == false
177   @tag assume_completion
178   @post __post._allowed[msg.sender][_spender] == _value
179 */
180 function approve(address _spender, uint256 _value) public onlyUnlocked whenNotPaused
     returns
181 (bool) {
182     return super.approve(_spender, _value);
183 }

```

```

184
185 //@CTK NO_OVERFLOW
186 //@CTK NO_BUF_OVERFLOW
187 //@CTK NO_ASF
188 /*@CTK "increaseApproval correctness"
189   @tag assume_completion
190   @post _paused == false
191   @post isLocked[msg.sender] == false
192   @post __post._allowed[msg.sender][_spender] == _allowed[msg.sender][_spender] +
       _addedValue
193 */
194 function increaseAllowance(address _spender, uint _addedValue) public onlyUnlocked
       whenNotPaused
195 returns (bool) {
196   return super.increaseAllowance(_spender, _addedValue);
197 }
198
199 //@CTK NO_OVERFLOW
200 //@CTK NO_BUF_OVERFLOW
201 //@CTK NO_ASF
202 /*@CTK decreaseApproval0
203   @pre __return == true
204   @pre _allowed[msg.sender][_spender] <= _subtractedValue
205   @post _paused == false
206   @post isLocked[msg.sender] == false
207   @post __post._allowed[msg.sender][_spender] == 0
208 */
209 /*@CTK decreaseApproval
210   @pre __return == true
211   @pre _allowed[msg.sender][_spender] > _subtractedValue
212   @post _paused == false
213   @post isLocked[msg.sender] == false
214   @post __post._allowed[msg.sender][_spender] ==
       _allowed[msg.sender][_spender] - _subtractedValue
215 */
216 */
217 function decreaseAllowance(address _spender, uint _subtractedValue) public
       onlyUnlocked
218 whenNotPaused returns
219 (bool) {
220   return super.decreaseAllowance(_spender, _subtractedValue);
221 }
222
223 function approveAndCall(address _spender, uint256 _value, bytes _extraData)
       onlyUnlocked whenNotPaused
224 public returns
225 (bool success) {
226   return super.approveAndCall(_spender, _value, _extraData);
227 }
228 }

```

File SafeMath.sol

```

1 pragma solidity ^0.4.24;
2
3 /**
4  * @title SafeMath
5  * @dev Unsigned math operations with safety checks that revert on error.
6  */
7 library SafeMath {

```

```

8  /**
9   * @dev Multiplies two unsigned integers, reverts on overflow.
10  */
11  /*@CTK "SafeMath mul"
12   @post (((a) > (0)) && (((a) * (b)) / (a)) != (b))) == (__reverted)
13   @post !__reverted -> __return == a * b
14   @post !__reverted == !__has_overflow
15   @post !(__has_buf_overflow)
16   @post !(__has_assertion_failure)
17  */
18  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
19      // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
20      // benefit is lost if 'b' is also tested.
21      // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
22      if (a == 0) {
23          return 0;
24      }
25
26      uint256 c = a * b;
27      require(c / a == b);
28
29      return c;
30  }
31
32  /**
33   * @dev Integer division of two unsigned integers truncating the quotient, reverts
34   * on division by zero.
35  */
36  /*@CTK "SafeMath div"
37   @post b != 0 -> !__reverted
38   @post !__reverted -> __return == a / b
39   @post !__reverted -> !__has_overflow
40   @post !(__has_buf_overflow)
41   @post !(__has_assertion_failure)
42  */
43  function div(uint256 a, uint256 b) internal pure returns (uint256) {
44      // Solidity only automatically asserts when dividing by 0
45      require(b > 0);
46      uint256 c = a / b;
47      // assert(a == b * c + a % b); // There is no case in which this doesn't hold
48
49      return c;
50  }
51
52  /**
53   * @dev Subtracts two unsigned integers, reverts on overflow (i.e. if subtrahend is
54   * greater than minuend).
55  */
56  /*@CTK "SafeMath sub"
57   @post (a < b) == __reverted
58   @post !__reverted -> __return == a - b
59   @post !__reverted -> !__has_overflow
60   @post !(__has_buf_overflow)
61   @post !(__has_assertion_failure)
62  */
63  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
64      require(b <= a);
65      uint256 c = a - b;

```

```

64
65     return c;
66 }
67
68 /**
69  * @dev Adds two unsigned integers, reverts on overflow.
70  */
71 /*@CTK "SafeMath add"
72  @post (a + b < a || a + b < b) == __reverted
73  @post !__reverted -> __return == a + b
74  @post !__reverted -> !__has_overflow
75  @post !(__has_buf_overflow)
76  @post !(__has_assertion_failure)
77 */
78 function add(uint256 a, uint256 b) internal pure returns (uint256) {
79     uint256 c = a + b;
80     require(c >= a);
81
82     return c;
83 }
84
85 /**
86  * @dev Divides two unsigned integers and returns the remainder (unsigned integer
87     modulo),
88  * reverts when dividing by zero.
89  */
90 /*@CTK "SafeMath div"
91  @post b != 0 -> !__reverted
92  @post !__reverted -> __return == a % b
93  @post !__reverted -> !__has_overflow
94  @post !(__has_buf_overflow)
95  @post !(__has_assertion_failure)
96 */
97 function mod(uint256 a, uint256 b) internal pure returns (uint256) {
98     require(b != 0);
99     return a % b;
100 }

```