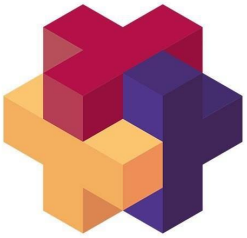
CertiK Audit Report For ITAMToken



Request Date: 2019-07-12
Revision Date: 2019-07-24
Platform Name: Ethereum

CERTIK

Contents

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 ITAMToken(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/

Exective Summary

This report has been prepared as the product of the Smart Contract Audit request by ITAMToken. This audit was conducted to discover issues and vulnerabilities in the source code of ITAMToken'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 vulnerabilies, but no concern found yet.

Testing Summary

PASS

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

Jul 24, 2019

Type of Issues

CertiK smart label engine applied 100% coveraged 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.	0	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

- **ITAMToken.sol** commit $bf48f308b7db3256b5a332a07719be8fecc1fcf3$
  `2300c97a9b048e6744b7e6832d81a303ad10ba6ba3ce9e6c708004d9b0043cf8`

Summary

CertiK was chosen by ITAM Games to audit the design and implementation of its soon to be released ITAMToken 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.

Discussions

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. Entries are labeled CRITICAL, IMPORTANT, INFO, and DISCUSSION (in a decreasing significance level manner).

ITAMToken.sol commit $96f7c9b7d5426c24526d352dace375776fab1c16$, previous

- CRITICAL `setAddresses()`: The update of the address (e.g. `_strategicSaleAddress`, `_privateSaleAddress`) happens only when the address is not `address(0)`, whereas the addresses are all initially `address(0)`.
  - (ITAM Games - Resolved) See latest commit $bf48f308b7db3256b5a332a07719be8fecc1fcf3$.
- INFO `modifier onlyNotBlackList` can be simplify to `require(!blackLists[msg.sender], ...)`
  - (ITAM Games - Resolved) See latest commit $bf48f308b7db3256b5a332a07719be8fecc1fcf3$.
- INFO `createOrUpdateItem()`: Recommend declaring the variables: `uint64 itemId`,`address tokenAddress`, `uint256 value` outside the for loop when gas efficiency is under consideration.
  - (ITAM Games - Resolved) See latest commit $bf48f308b7db3256b5a332a07719be8fecc1fcf3$.

- INFO `purchaseItemOnITAM()`: Recommend adding `onlyNotBlackList` modifier.
  - (ITAM Games - Resolved) See latest commit $bf48f308b7db3256b5a332a07719be8fecc1fcf3$.
- INFO `setBlackList` is mixing all the control into a single function. Would it be better to split it into `addToBlackList`, `removeFromBlackList`, `isBlackList`. Recommend emitting log for monitoring the blacklist update.
  - (ITAM Games - Resolved) See latest commit $bf48f308b7db3256b5a332a07719be8fecc1fcf3$.
- IMPORTANT Recommend specifing the default `payable` callback function.
  - (ITAM Games - Resolved) See latest commit $bf48f308b7db3256b5a332a07719be8fecc1fcf3$.
- IMPORTANT `createOrUpdateItem()`, `deleteItems()`: No `bool` result returned.
  - (ITAM Games - Resolved) See latest commit $bf48f308b7db3256b5a332a07719be8fecc1fcf3$.
- Function `resetPurchaseInAppDiscountInfo()`:
  - IMPORTANT Considier adding check `require(endTime > startTime, ...)`.
    - (ITAM Games - Resolved) See latest commit $bf48f308b7db3256b5a332a07719be8fecc1fcf3$.
  - INFO Consider adding error messages to `require()` checks.
    - (ITAM Games - Resolved) See latest commit $bf48f308b7db3256b5a332a07719be8fecc1fcf3$.
  - INFO Consdier declaring `startTime`, `endTime`, `percent` outside the for loop.
    - (ITAM Games - Resolved) See latest commit $bf48f308b7db3256b5a332a07719be8fecc1fcf3$.
- DISCUSSION `constructor()`: Recommend using OpenZeppelin's `Ownable` for ownership and corresponding access control.
- DISCUSSION `withdrawEther()`: Given `withdrawEther()` is an important function, highly recommend to evaluate and adapt to use the `Withdraw Pattern` from solidity tutorial to mimic the lost caused by simple error.

Static Analysis Results

INSECURE_COMPILER_VERSION

Line 3 in File ITAMToken.sol

```
3 pragma solidity ^0.5.2;
```

ℹ Only these compiler versions are safe to compile your code: 0.5.9

TIMESTAMP_DEPENDENCY

Line 744 in File ITAMToken.sol

```
744             if(discountInfo.startTime <= now) {
```

❗ ”now” can be influenced by minors to some degree

TIMESTAMP_DEPENDENCY

Line 745 in File ITAMToken.sol

```
745                 if(now <= discountInfo.endTime) {
```

❗ ”now” can be influenced by minors to some degree

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
53             balance: 0x0
54         }
55     }
56
```

Post environment

```
57 After Execution:
58     Input = {
59         from = 0x0
60         to = 0x0
61         tokens = 0x6c
```

Formal Verification Request 1

SafeMath mul

📅 24, Jul 2019
⏱ 346.57 ms

Line 13-18 in File ITAMToken.sol

```
13    /*@CTK "SafeMath mul"
14      @post (a > 0) && (((a * b) / a) != b) -> __reverted
15      @post __reverted -> (a > 0) && (((a * b) / a) != b)
16      @post !__reverted -> __return == a * b
17      @post !__reverted == !__has_overflow
18     */
```

Line 19-31 in File ITAMToken.sol

```
19    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
20        // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
21        // benefit is lost if 'b' is also tested.
22        // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
23        if (a == 0) {
24            return 0;
25        }
26
27        uint256 c = a * b;
28        require(c / a == b);
29
30        return c;
31    }
```

✅ The code meets the specification.

Formal Verification Request 2

SafeMath div

📅 24, Jul 2019
⏱ 15.37 ms

Line 36-40 in File ITAMToken.sol

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

Line 41-48 in File ITAMToken.sol

```
41    function div(uint256 a, uint256 b) internal pure returns (uint256) {
42        // Solidity only automatically asserts when dividing by 0
43        require(b > 0);
44        uint256 c = a / b;
45        // assert(a == b * c + a % b); // There is no case in which this doesn't hold
46
47        return c;
48    }
```

The code meets the specification.

Formal Verification Request 3

SafeMath sub

24, Jul 2019
14.15 ms

Line 53-57 in File ITAMToken.sol

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

Line 58-63 in File ITAMToken.sol

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

The code meets the specification.

Formal Verification Request 4

SafeMath add

24, Jul 2019
17.68 ms

Line 68-72 in File ITAMToken.sol

```
68    /*@CTK "SafeMath add"
69      @post (a + b < a || a + b < b) == __reverted
70      @post !__reverted -> __return == a + b
71      @post !__reverted -> !__has_overflow
72     */
```

Line 73-78 in File ITAMToken.sol

```
73    function add(uint256 a, uint256 b) internal pure returns (uint256) {
74        uint256 c = a + b;
75        require(c >= a);
76
77        return c;
78    }
```

The code meets the specification.

Formal Verification Request 5

SafeMath mod

📅 24, Jul 2019
⏱ 14.61 ms

Line 84-89 in File ITAMToken.sol

```
84    /*@CTK "SafeMath mod"
85      @post (b == 0) == __reverted
86      @post !__reverted -> b != 0
87      @post !__reverted -> __return == a % b
88      @post !__reverted -> !__has_overflow
89     */
```

Line 90-93 in File ITAMToken.sol

```
90    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
91        require(b != 0);
92        return a % b;
93    }
```

✅ The code meets the specification.

Formal Verification Request 6

totalSupply

📅 24, Jul 2019
⏱ 6.43 ms

Line 152-154 in File ITAMToken.sol

```
152   /*@CTK totalSupply
153     @post __return == _totalSupply
154    */
```

Line 155-157 in File ITAMToken.sol

```
155   function totalSupply() public view returns (uint256) {
156       return _totalSupply;
157   }
```

✅ The code meets the specification.

Formal Verification Request 7

balanceOf

📅 24, Jul 2019
⏱ 6.04 ms

Line 164-166 in File ITAMToken.sol

```
164   /*@CTK balanceOf
165     @post __return == _balances[owner]
166    */
```

Line 167-169 in File ITAMToken.sol

```
167     function balanceOf(address owner) public view returns (uint256) {
168         return _balances[owner];
169     }
```

The code meets the specification.

Formal Verification Request 8

allowance

24, Jul 2019
6.95 ms

Line 177-179 in File ITAMToken.sol

```
177     /*@CTK allowance
178       @post __return == _allowed[owner][spender]
179      */
```

Line 180-182 in File ITAMToken.sol

```
180     function allowance(address owner, address spender) public view returns (uint256) {
181         return _allowed[owner][spender];
182     }
```

The code meets the specification.

Formal Verification Request 9

transfer

24, Jul 2019
190.64 ms

Line 189-196 in File ITAMToken.sol

```
189     /*@CTK transfer
190       @tag assume_completion
191       @pre msg.sender != to
192       @post to != address(0)
193       @post value <= _balances[msg.sender]
194       @post __post._balances[to] == _balances[to] + value
195       @post __post._balances[msg.sender] == _balances[msg.sender] - value
196      */
```

Line 197-200 in File ITAMToken.sol

```
197     function transfer(address to, uint256 value) public returns (bool) {
198         _transfer(msg.sender, to, value);
199         return true;
200     }
```

The code meets the specification.

Formal Verification Request 10

approve

24, Jul 2019
55.68 ms

Line 211-215 in File ITAMToken.sol

```
211     /*@CTK approve
212       @tag assume_completion
213       @post spender != address(0)
214       @post __post._allowed[msg.sender][spender] == value
215      */
```

Line 216-219 in File ITAMToken.sol

```
216     function approve(address spender, uint256 value) public returns (bool) {
217         _approve(msg.sender, spender, value);
218         return true;
219     }
```

The code meets the specification.

Formal Verification Request 11

transfer_from

24, Jul 2019
306.09 ms

Line 229-238 in File ITAMToken.sol

```
229     /*@CTK transfer_from
230       @tag assume_completion
231       @pre from != to
232       @post to != address(0)
233       @post value <= _allowed[from][msg.sender]
234       @post __post._balances[from] == _balances[from] - value
235       @post __post._balances[to] == _balances[to] + value
236       @post __post._allowed[from][msg.sender] ==
237       _allowed[from][msg.sender] - value
238      */
```

Line 239-243 in File ITAMToken.sol

```
239     function transferFrom(address from, address to, uint256 value) public returns (
            bool) {
240         _transfer(from, to, value);
241         _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));
242         return true;
243     }
```

The code meets the specification.

Formal Verification Request 12

mint

24, Jul 2019
81.65 ms

Line 266-271 in File ITAMToken.sol

```
266     /*@CTK mint
267       @tag assume_completion
268       @post account != 0
269       @post __post._totalSupply == _totalSupply + value
270       @post __post._balances[account] == _balances[account] + value
271      */
```

Line 272-278 in File ITAMToken.sol

```
272     function mint(address account, uint256 value) internal {
273         require(account != address(0));
274 
275         _totalSupply = _totalSupply.add(value);
276         _balances[account] = _balances[account].add(value);
277         emit Transfer(address(0), account, value);
278     }
```

The code meets the specification.

Formal Verification Request 13

ERC20Capped

24, Jul 2019
15.37 ms

Line 313-317 in File ITAMToken.sol

```
313     /*@CTK ERC20Capped
314       @tag assume_completion
315       @post cap > 0
316       @post __post._cap == cap
317      */
```

Line 318-321 in File ITAMToken.sol

```
318     constructor (uint256 cap) public {
319         require(cap > 0);
320         _cap = cap;
321     }
```

The code meets the specification.

Formal Verification Request 14

cap

24, Jul 2019
6.38 ms

Line 326-328 in File ITAMToken.sol

```
326     /*@CTK cap
327       @post __return == _cap
328      */
```

Line 329-331 in File ITAMToken.sol

```
329     function cap() public view returns (uint256) {
330         return _cap;
331     }
```

The code meets the specification.

Formal Verification Request 15

_mint

24, Jul 2019
351.44 ms

Line 333-339 in File ITAMToken.sol

```
333     /*@CTK _mint
334       @tag assume_completion
335       @post _totalSupply + value <= _cap
336       @post account != address(0)
337       @post __post._totalSupply == _totalSupply + value
338       @post __post._balances[account] == _balances[account] + value
339      */
```

Line 340-343 in File ITAMToken.sol

```
340     function _mint(address account, uint256 value) internal {
341         require(totalSupply().add(value) <= _cap);
342         super.mint(account, value);
343     }
```

The code meets the specification.

Formal Verification Request 16

If method completes, integer overflow would not happen.

24, Jul 2019
125.93 ms

Line 419 in File ITAMToken.sol

```
419     //@CTK NO_OVERFLOW
```

Line 435-447 in File ITAMToken.sol

```
435     constructor(address _owner, address _gameMaster, address _strategicSaleAddress,
            address _privateSaleAddress, address _publicSaleAddress, address _teamAddress,
             address _advisorAddress, address _marketingAddress, address _ecoAddress,
436                 address payable _inAppAddress) public ERC20Capped(TOTAL_CAP) {
```

```
437         owner = _owner;
438         gameMaster = _gameMaster;
439         strategicSaleAddress = _strategicSaleAddress;
440         privateSaleAddress = _privateSaleAddress;
441         publicSaleAddress = _publicSaleAddress;
442         teamAddress = _teamAddress;
443         advisorAddress = _advisorAddress;
444         marketingAddress = _marketingAddress;
445         ecoAddress = _ecoAddress;
446         inAppAddress = _inAppAddress;
447     }
```

The code meets the specification.

Formal Verification Request 17

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

24, Jul 2019
1.2 ms

Line 420 in File ITAMToken.sol

```
420     //@CTK NO_BUF_OVERFLOW
```

Line 435-447 in File ITAMToken.sol

```
435     constructor(address _owner, address _gameMaster, address _strategicSaleAddress,
            address _privateSaleAddress, address _publicSaleAddress, address _teamAddress,
             address _advisorAddress, address _marketingAddress, address _ecoAddress,
436                 address payable _inAppAddress) public ERC20Capped(TOTAL_CAP) {
437         owner = _owner;
438         gameMaster = _gameMaster;
439         strategicSaleAddress = _strategicSaleAddress;
440         privateSaleAddress = _privateSaleAddress;
441         publicSaleAddress = _publicSaleAddress;
442         teamAddress = _teamAddress;
443         advisorAddress = _advisorAddress;
444         marketingAddress = _marketingAddress;
445         ecoAddress = _ecoAddress;
446         inAppAddress = _inAppAddress;
447     }
```

The code meets the specification.

Formal Verification Request 18

Method will not encounter an assertion failure.

24, Jul 2019
1.1 ms

Line 421 in File ITAMToken.sol

```
421     //@CTK NO_ASF
```

Line 435-447 in File ITAMToken.sol

```
435     constructor(address _owner, address _gameMaster, address _strategicSaleAddress,
            address _privateSaleAddress, address _publicSaleAddress, address _teamAddress,
             address _advisorAddress, address _marketingAddress, address _ecoAddress,
436                 address payable _inAppAddress) public ERC20Capped(TOTAL_CAP) {
437         owner = _owner;
438         gameMaster = _gameMaster;
439         strategicSaleAddress = _strategicSaleAddress;
440         privateSaleAddress = _privateSaleAddress;
441         publicSaleAddress = _publicSaleAddress;
442         teamAddress = _teamAddress;
443         advisorAddress = _advisorAddress;
444         marketingAddress = _marketingAddress;
445         ecoAddress = _ecoAddress;
446         inAppAddress = _inAppAddress;
447     }
```

The code meets the specification.

Formal Verification Request 19

ITAMToken constructor

24, Jul 2019
2.9 ms

Line 422-434 in File ITAMToken.sol

```
422     /*@CTK "ITAMToken constructor"
423       @tag assume_completion
424       @post __post.owner == _owner
425       @post __post.gameMaster == _gameMaster
426       @post __post.strategicSaleAddress == _strategicSaleAddress
427       @post __post.privateSaleAddress == _privateSaleAddress
428       @post __post.publicSaleAddress == _publicSaleAddress
429       @post __post.teamAddress == _teamAddress
430       @post __post.advisorAddress == _advisorAddress
431       @post __post.marketingAddress == _marketingAddress
432       @post __post.ecoAddress == _ecoAddress
433       @post __post.inAppAddress == _inAppAddress
434      */
```

Line 435-447 in File ITAMToken.sol

```
435     constructor(address _owner, address _gameMaster, address _strategicSaleAddress,
            address _privateSaleAddress, address _publicSaleAddress, address _teamAddress,
             address _advisorAddress, address _marketingAddress, address _ecoAddress,
436                 address payable _inAppAddress) public ERC20Capped(TOTAL_CAP) {
437         owner = _owner;
438         gameMaster = _gameMaster;
439         strategicSaleAddress = _strategicSaleAddress;
440         privateSaleAddress = _privateSaleAddress;
441         publicSaleAddress = _publicSaleAddress;
442         teamAddress = _teamAddress;
443         advisorAddress = _advisorAddress;
444         marketingAddress = _marketingAddress;
445         ecoAddress = _ecoAddress;
446         inAppAddress = _inAppAddress;
447     }
```

The code meets the specification.

Formal Verification Request 20

If method completes, integer overflow would not happen.

24, Jul 2019
19.31 ms

Line 459 in File ITAMToken.sol

```
459     //@CTK NO_OVERFLOW
```

Line 466-468 in File ITAMToken.sol

```
466     function setGameMaster(address _gameMaster) public onlyOwner {
467         gameMaster = _gameMaster;
468     }
```

The code meets the specification.

Formal Verification Request 21

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

24, Jul 2019
0.46 ms

Line 460 in File ITAMToken.sol

```
460     //@CTK NO_BUF_OVERFLOW
```

Line 466-468 in File ITAMToken.sol

```
466     function setGameMaster(address _gameMaster) public onlyOwner {
467         gameMaster = _gameMaster;
468     }
```

The code meets the specification.

Formal Verification Request 22

Method will not encounter an assertion failure.

24, Jul 2019
0.43 ms

Line 461 in File ITAMToken.sol

```
461     //@CTK NO_ASF
```

Line 466-468 in File ITAMToken.sol

```
466     function setGameMaster(address _gameMaster) public onlyOwner {
467         gameMaster = _gameMaster;
468     }
```

The code meets the specification.

Formal Verification Request 23

setGameMaster

24, Jul 2019
4.52 ms

Line 462-465 in File ITAMToken.sol

```
462     /*@CTK setGameMaster
463       @tag assume_completion
464       @post __post.gameMaster == _gameMaster
465      */
```

Line 466-468 in File ITAMToken.sol

```
466     function setGameMaster(address _gameMaster) public onlyOwner {
467         gameMaster = _gameMaster;
468     }
```

The code meets the specification.

Formal Verification Request 24

If method completes, integer overflow would not happen.

24, Jul 2019
423.7 ms

Line 470 in File ITAMToken.sol

```
470     //@CTK NO_OVERFLOW
```

Line 482-484 in File ITAMToken.sol

```
482     function transfer(address _to, uint256 _value) public onlyNotBlackList returns (
            bool) {
483         return super.transfer(_to, _value);
484     }
```

The code meets the specification.

Formal Verification Request 25

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

24, Jul 2019
38.03 ms

Line 471 in File ITAMToken.sol

```
471     //@CTK NO_BUF_OVERFLOW
```

Line 482-484 in File ITAMToken.sol

```
482     function transfer(address _to, uint256 _value) public onlyNotBlackList returns (
            bool) {
483         return super.transfer(_to, _value);
484     }
```

The code meets the specification.

Formal Verification Request 26

Method will not encounter an assertion failure.

24, Jul 2019
35.46 ms

Line 472 in File ITAMToken.sol

```
472     //@CTK NO_ASF
```

Line 482-484 in File ITAMToken.sol

```
482     function transfer(address _to, uint256 _value) public onlyNotBlackList returns (
            bool) {
483         return super.transfer(_to, _value);
484     }
```

The code meets the specification.

Formal Verification Request 27

transfer correctness

24, Jul 2019
448.22 ms

Line 473-481 in File ITAMToken.sol

```
473     /*@CTK "transfer correctness"
474       @tag assume_completion
475       @post blackLists[msg.sender] == false
476       @post _to != 0x0
477       @post _value <= _balances[msg.sender]
478       @post _to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
              - _value
479       @post _to != msg.sender -> __post._balances[_to] == _balances[_to] + _value
480       @post _to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
481      */
```

Line 482-484 in File ITAMToken.sol

```
482     function transfer(address _to, uint256 _value) public onlyNotBlackList returns (
            bool) {
483         return super.transfer(_to, _value);
484     }
```

The code meets the specification.

Formal Verification Request 28

If method completes, integer overflow would not happen.

24, Jul 2019
577.14 ms

Line 486 in File ITAMToken.sol

```
486     //@CTK NO_OVERFLOW
```

Line 499-501 in File ITAMToken.sol

```
499     function transferFrom(address _from, address _to, uint256 _value) public
            onlyNotBlackList returns (bool) {
500         return super.transferFrom(_from, _to, _value);
501     }
```

The code meets the specification.

Formal Verification Request 29

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

24, Jul 2019
118.33 ms

Line 487 in File ITAMToken.sol

```
487     //@CTK NO_BUF_OVERFLOW
```

Line 499-501 in File ITAMToken.sol

```
499     function transferFrom(address _from, address _to, uint256 _value) public
            onlyNotBlackList returns (bool) {
500         return super.transferFrom(_from, _to, _value);
501     }
```

The code meets the specification.

Formal Verification Request 30

Method will not encounter an assertion failure.

24, Jul 2019
118.65 ms

Line 488 in File ITAMToken.sol

```
488     //@CTK NO_ASF
```

Line 499-501 in File ITAMToken.sol

```
499     function transferFrom(address _from, address _to, uint256 _value) public
            onlyNotBlackList returns (bool) {
500         return super.transferFrom(_from, _to, _value);
501     }
```

The code meets the specification.

Formal Verification Request 31

transferFrom correctness

24, Jul 2019
1060.89 ms

Line 489-498 in File ITAMToken.sol

```
489     /*@CTK "transferFrom correctness"
490       @tag assume_completion
491       @post blackLists[msg.sender] == false
492       @post _to != 0x0
493       @post _value <= _balances[_from] && _value <= _allowed[_from][msg.sender]
494       @post _to != _from -> __post._balances[_from] == _balances[_from] - _value
495       @post _to != _from -> __post._balances[_to] == _balances[_to] + _value
496       @post _to == _from -> __post._balances[_from] == _balances[_from]
497       @post __post._allowed[_from][msg.sender] == _allowed[_from][msg.sender] - _value
498      */
```

Line 499-501 in File ITAMToken.sol

```
499     function transferFrom(address _from, address _to, uint256 _value) public
            onlyNotBlackList returns (bool) {
500         return super.transferFrom(_from, _to, _value);
501     }
```

The code meets the specification.

Formal Verification Request 32

If method completes, integer overflow would not happen.

24, Jul 2019
123.49 ms

Line 503 in File ITAMToken.sol

```
503     //@CTK NO_OVERFLOW
```

Line 512-514 in File ITAMToken.sol

```
512     function approve(address spender, uint256 value) public onlyNotBlackList returns (
            bool) {
513         return super.approve(spender, value);
514     }
```

The code meets the specification.

Formal Verification Request 33

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

24, Jul 2019
1.26 ms

Line 504 in File ITAMToken.sol

```
504     //@CTK NO_BUF_OVERFLOW
```

Line 512-514 in File ITAMToken.sol

```
512     function approve(address spender, uint256 value) public onlyNotBlackList returns (
            bool) {
513         return super.approve(spender, value);
514     }
```

The code meets the specification.

Formal Verification Request 34

Method will not encounter an assertion failure.

24, Jul 2019
1.21 ms

Line 505 in File ITAMToken.sol

```
505     //@CTK NO_ASF
```

Line 512-514 in File ITAMToken.sol

```
512     function approve(address spender, uint256 value) public onlyNotBlackList returns (
            bool) {
513         return super.approve(spender, value);
514     }
```

The code meets the specification.

Formal Verification Request 35

approve correctness

24, Jul 2019
34.56 ms

Line 506-511 in File ITAMToken.sol

```
506     /*@CTK "approve correctness"
507       @tag assume_completion
508       @post blackLists[msg.sender] == false
509       @post spender != address(0)
510       @post __post._allowed[msg.sender][spender] == value
511      */
```

Line 512-514 in File ITAMToken.sol

```
512     function approve(address spender, uint256 value) public onlyNotBlackList returns (
            bool) {
513         return super.approve(spender, value);
514     }
```

The code meets the specification.

Formal Verification Request 36

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

24, Jul 2019
248.4 ms

Line 516 in File ITAMToken.sol

```
516     //@CTK NO_BUF_OVERFLOW
```

Line 524-526 in File ITAMToken.sol

```
524     function burn(uint256 value) public onlyOwner {
525         super._burn(msg.sender, value);
526     }
```

The code meets the specification.

Formal Verification Request 37

Method will not encounter an assertion failure.

24, Jul 2019
40.68 ms

Line 517 in File ITAMToken.sol

```
517     //@CTK NO_ASF
```

Line 524-526 in File ITAMToken.sol

```
524     function burn(uint256 value) public onlyOwner {
525         super._burn(msg.sender, value);
526     }
```

The code meets the specification.

Formal Verification Request 38

burn correctness

24, Jul 2019
427.24 ms

Line 518-523 in File ITAMToken.sol

```
518     /*@CTK "burn correctness"
519       @tag assume_completion
520       @post msg.sender == owner
521       @post __post._totalSupply == _totalSupply - value
522       @post __post._balances[msg.sender] == _balances[msg.sender] - value
523      */
```

Line 524-526 in File ITAMToken.sol

```
524     function burn(uint256 value) public onlyOwner {
525         super._burn(msg.sender, value);
526     }
```

The code meets the specification.

Formal Verification Request 39

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

24, Jul 2019
4041.95 ms

Line 529 in File ITAMToken.sol

```
529     //@CTK NO_BUF_OVERFLOW
```

Line 553-586 in File ITAMToken.sol

```
553     function unlock() public onlyOwner returns (bool) {
554         uint8 _unlockCount = unlockCount;
555 
556         if(strategicSaleReleaseCaps.length > _unlockCount) {
557             super._mint(strategicSaleAddress, strategicSaleReleaseCaps[_unlockCount]);
558         }
559 
560         if(privateSaleReleaseCaps.length > _unlockCount) {
561             super._mint(privateSaleAddress, privateSaleReleaseCaps[_unlockCount]);
562         }
563 
564         if(_unlockCount == 0) {
565             super._mint(publicSaleAddress, publicSaleReleaseCap);
566         }
567 
568         if(teamReleaseCaps.length > _unlockCount) {
569             super._mint(teamAddress, teamReleaseCaps[_unlockCount]);
570         }
571 
572         if(advisorReleaseCaps.length > _unlockCount) {
573             super._mint(advisorAddress, advisorReleaseCaps[_unlockCount]);
574         }
575 
576         if(marketingReleaseCaps.length > _unlockCount) {
577             super._mint(marketingAddress, marketingReleaseCaps[_unlockCount]);
578         }
579 
580         if(ecoReleaseCaps.length > _unlockCount) {
581             super._mint(ecoAddress, ecoReleaseCaps[_unlockCount]);
582         }
583 
584         unlockCount++;
585         return true;
586     }
```

The code meets the specification.

Formal Verification Request 40

Method will not encounter an assertion failure.

24, Jul 2019

2115.99 ms

Line 530 in File ITAMToken.sol

```
530     //@CTK NO_ASF
```

Line 553-586 in File ITAMToken.sol

```
553     function unlock() public onlyOwner returns (bool) {
554         uint8 _unlockCount = unlockCount;
555 
556         if(strategicSaleReleaseCaps.length > _unlockCount) {
557             super._mint(strategicSaleAddress, strategicSaleReleaseCaps[_unlockCount]);
558         }
559 
560         if(privateSaleReleaseCaps.length > _unlockCount) {
561             super._mint(privateSaleAddress, privateSaleReleaseCaps[_unlockCount]);
562         }
563 
564         if(_unlockCount == 0) {
565             super._mint(publicSaleAddress, publicSaleReleaseCap);
566         }
567 
568         if(teamReleaseCaps.length > _unlockCount) {
569             super._mint(teamAddress, teamReleaseCaps[_unlockCount]);
570         }
571 
572         if(advisorReleaseCaps.length > _unlockCount) {
573             super._mint(advisorAddress, advisorReleaseCaps[_unlockCount]);
574         }
575 
576         if(marketingReleaseCaps.length > _unlockCount) {
577             super._mint(marketingAddress, marketingReleaseCaps[_unlockCount]);
578         }
579 
580         if(ecoReleaseCaps.length > _unlockCount) {
581             super._mint(ecoAddress, ecoReleaseCaps[_unlockCount]);
582         }
583 
584         unlockCount++;
585         return true;
586     }
```

The code meets the specification.

Formal Verification Request 41

If method completes, integer overflow would not happen.

24, Jul 2019
88.92 ms

Line 588 in File ITAMToken.sol

```
588     //@CTK NO_OVERFLOW
```

Line 603-613 in File ITAMToken.sol

```
603     function setAddresses(address _strategicSaleAddress, address _privateSaleAddress,
            address _publicSaleAddress, address _teamAddress, address _advisorAddress,
            address _marketingAddress, address _ecoAddress,
604                         address payable _inAppAddress) public onlyOwner {
605         strategicSaleAddress = _strategicSaleAddress;
606         privateSaleAddress = _privateSaleAddress;
607         publicSaleAddress = _publicSaleAddress;
608         teamAddress = _teamAddress;
609         advisorAddress = _advisorAddress;
610         marketingAddress = _marketingAddress;
611         ecoAddress = _ecoAddress;
612         inAppAddress = _inAppAddress;
613     }
```

The code meets the specification.

Formal Verification Request 42

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

24, Jul 2019
0.97 ms

Line 589 in File ITAMToken.sol

```
589     //@CTK NO_BUF_OVERFLOW
```

Line 603-613 in File ITAMToken.sol

```
603     function setAddresses(address _strategicSaleAddress, address _privateSaleAddress,
            address _publicSaleAddress, address _teamAddress, address _advisorAddress,
            address _marketingAddress, address _ecoAddress,
604                         address payable _inAppAddress) public onlyOwner {
605         strategicSaleAddress = _strategicSaleAddress;
606         privateSaleAddress = _privateSaleAddress;
607         publicSaleAddress = _publicSaleAddress;
608         teamAddress = _teamAddress;
609         advisorAddress = _advisorAddress;
610         marketingAddress = _marketingAddress;
611         ecoAddress = _ecoAddress;
612         inAppAddress = _inAppAddress;
613     }
```

The code meets the specification.

Formal Verification Request 43

Method will not encounter an assertion failure.

24, Jul 2019
0.93 ms

Line 590 in File ITAMToken.sol

```
590     //@CTK NO_ASF
```

Line 603-613 in File ITAMToken.sol

```
603     function setAddresses(address _strategicSaleAddress, address _privateSaleAddress,
            address _publicSaleAddress, address _teamAddress, address _advisorAddress,
            address _marketingAddress, address _ecoAddress,
604                         address payable _inAppAddress) public onlyOwner {
605         strategicSaleAddress = _strategicSaleAddress;
606         privateSaleAddress = _privateSaleAddress;
607         publicSaleAddress = _publicSaleAddress;
608         teamAddress = _teamAddress;
609         advisorAddress = _advisorAddress;
610         marketingAddress = _marketingAddress;
611         ecoAddress = _ecoAddress;
612         inAppAddress = _inAppAddress;
613     }
```

The code meets the specification.

Formal Verification Request 44

setAddresses

24, Jul 2019
32.54 ms

Line 591-602 in File ITAMToken.sol

```
591     /*@CTK setAddresses
592       @tag assume_completion
593       @post msg.sender == owner
594       @post __post.strategicSaleAddress == _strategicSaleAddress
595       @post __post.privateSaleAddress == _privateSaleAddress
596       @post __post.publicSaleAddress == _publicSaleAddress
597       @post __post.teamAddress == _teamAddress
598       @post __post.advisorAddress == _advisorAddress
599       @post __post.marketingAddress == _marketingAddress
600       @post __post.ecoAddress == _ecoAddress
601       @post __post.inAppAddress == _inAppAddress
602      */
```

Line 603-613 in File ITAMToken.sol

```
603     function setAddresses(address _strategicSaleAddress, address _privateSaleAddress,
            address _publicSaleAddress, address _teamAddress, address _advisorAddress,
            address _marketingAddress, address _ecoAddress,
604                         address payable _inAppAddress) public onlyOwner {
605         strategicSaleAddress = _strategicSaleAddress;
606         privateSaleAddress = _privateSaleAddress;
607         publicSaleAddress = _publicSaleAddress;
608         teamAddress = _teamAddress;
609         advisorAddress = _advisorAddress;
610         marketingAddress = _marketingAddress;
611         ecoAddress = _ecoAddress;
612         inAppAddress = _inAppAddress;
613     }
```

The code meets the specification.

Formal Verification Request 45

If method completes, integer overflow would not happen.

24, Jul 2019
36.31 ms

Line 615 in File ITAMToken.sol

```
615     //@CTK NO_OVERFLOW
```

Line 624-627 in File ITAMToken.sol

```
624     function addToBlackList(address _to) public onlyOwner {
625         require(!blackLists[_to], "already blacklist");
626         blackLists[_to] = true;
627     }
```

The code meets the specification.

Formal Verification Request 46

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

24, Jul 2019
0.86 ms

Line 616 in File ITAMToken.sol

```
616     //@CTK NO_BUF_OVERFLOW
```

Line 624-627 in File ITAMToken.sol

```
624     function addToBlackList(address _to) public onlyOwner {
625         require(!blackLists[_to], "already blacklist");
626         blackLists[_to] = true;
627     }
```

The code meets the specification.

Formal Verification Request 47

Method will not encounter an assertion failure.

24, Jul 2019
0.73 ms

Line 617 in File ITAMToken.sol

```
617     //@CTK NO_ASF
```

Line 624-627 in File ITAMToken.sol

```
624     function addToBlackList(address _to) public onlyOwner {
625         require(!blackLists[_to], "already blacklist");
626         blackLists[_to] = true;
627     }
```

The code meets the specification.

Formal Verification Request 48

addToBlackList

24, Jul 2019
10.27 ms

Line 618-623 in File ITAMToken.sol

```
618     /*@CTK addToBlackList
619       @tag assume_completion
620       @post msg.sender == owner
621       @post blackLists[_to] == false
622       @post __post.blackLists[_to] == true
623      */
```

Line 624-627 in File ITAMToken.sol

```
624     function addToBlackList(address _to) public onlyOwner {
625         require(!blackLists[_to], "already blacklist");
626         blackLists[_to] = true;
627     }
```

The code meets the specification.

Formal Verification Request 49

If method completes, integer overflow would not happen.

24, Jul 2019
44.75 ms

Line 629 in File ITAMToken.sol

```
629     //@CTK NO_OVERFLOW
```

Line 638-641 in File ITAMToken.sol

```
638     function removeFromBlackList(address _to) public onlyOwner {
639         require(blackLists[_to], "cannot found this address from blacklist");
640         blackLists[_to] = false;
641     }
```

The code meets the specification.

Formal Verification Request 50

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

24, Jul 2019
0.83 ms

Line 630 in File ITAMToken.sol

```
630     //@CTK NO_BUF_OVERFLOW
```

Line 638-641 in File ITAMToken.sol

```
638     function removeFromBlackList(address _to) public onlyOwner {
639         require(blackLists[_to], "cannot found this address from blacklist");
640         blackLists[_to] = false;
641     }
```

The code meets the specification.

Formal Verification Request 51

Method will not encounter an assertion failure.

24, Jul 2019
0.75 ms

Line 631 in File ITAMToken.sol

```
631     //@CTK NO_ASF
```

Line 638-641 in File ITAMToken.sol

```
638     function removeFromBlackList(address _to) public onlyOwner {
639         require(blackLists[_to], "cannot found this address from blacklist");
640         blackLists[_to] = false;
641     }
```

The code meets the specification.

Formal Verification Request 52

removeFromBlackList

24, Jul 2019
10.09 ms

Line 632-637 in File ITAMToken.sol

```
632     /*@CTK removeFromBlackList
633       @tag assume_completion
634       @post msg.sender == owner
635       @post blackLists[_to] == true
636       @post __post.blackLists[_to] == false
637      */
```

Line 638-641 in File ITAMToken.sol

```
638     function removeFromBlackList(address _to) public onlyOwner {
639         require(blackLists[_to], "cannot found this address from blacklist");
640         blackLists[_to] = false;
641     }
```

The code meets the specification.

Formal Verification Request 53

If method completes, integer overflow would not happen.

24, Jul 2019
23.65 ms

Line 653 in File ITAMToken.sol

```
653     //@CTK NO_OVERFLOW
```

Line 659-662 in File ITAMToken.sol

```
659     function withdrawEther(uint256 amount) public onlyOwner {
660         inAppAddress.transfer(amount);
661         emit WithdrawEther(inAppAddress, amount);
662     }
```

The code meets the specification.

Formal Verification Request 54

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

24, Jul 2019
0.67 ms

Line 654 in File ITAMToken.sol

```
654     //@CTK NO_BUF_OVERFLOW
```

Line 659-662 in File ITAMToken.sol

```
659     function withdrawEther(uint256 amount) public onlyOwner {
660         inAppAddress.transfer(amount);
661         emit WithdrawEther(inAppAddress, amount);
662     }
```

The code meets the specification.

Formal Verification Request 55

withdrawEther correctness

24, Jul 2019
2.27 ms

Line 655-658 in File ITAMToken.sol

```
655     /*@CTK "withdrawEther correctness"
656       @tag assume_completion
657       @post msg.sender == owner
658      */
```

Line 659-662 in File ITAMToken.sol

```
659     function withdrawEther(uint256 amount) public onlyOwner {
660         inAppAddress.transfer(amount);
661         emit WithdrawEther(inAppAddress, amount);
662     }
```

The code meets the specification.

Formal Verification Request 56

createOrUpdateItem correctness

24, Jul 2019
55.8 ms

Line 664-670 in File ITAMToken.sol

```
664     /*@CTK "createOrUpdateItem correctness"
665       @tag assume_completion
666       @post msg.sender == gameMaster
667       @post itemIds.length == tokenAddresses.length
668       @post tokenAddresses.length == values.length
669       @post __return == true
670      */
```

Line 671-702 in File ITAMToken.sol

```
671     function createOrUpdateItem(uint64 appId, uint64[] memory itemIds, address[]
            memory tokenAddresses, uint256[] memory values) public onlyGameMaster returns(
            bool) {
672         uint itemLength = itemIds.length;
673         require(itemLength == tokenAddresses.length && tokenAddresses.length == values.
                length);
674 
675         uint64 itemId;
676         address tokenAddress;
677         uint256 value;
678         /*@CTK "forLoop in createOrUpdateItem"
679           @var uint64 appId
680           @var uint64[] itemIds
681           @var address[] tokenAddresses
682           @var uint256[] values
683           @var ITAMToken this
684           @var uint itemLength
685           @pre itemLength == itemIds.length
686           @pre itemLength == tokenAddresses.length
687           @pre itemLength == values.length
688           @inv i <= itemLength
689           @post i >= itemLength
690           @post !__should_return
691           */
692         for(uint16 i = 0; i < itemLength; i++) {
693             itemId = itemIds[i];
694             tokenAddress = tokenAddresses[i];
695             value = values[i];
696 
697             items[appId][itemId][tokenAddress] = value;
698         }
699 
```

```
700         emit SetItem(appId);
701         return true;
702     }
```

The code meets the specification.

Formal Verification Request 57

If method completes, integer overflow would not happen.

24, Jul 2019
290.13 ms

Line 710 in File ITAMToken.sol

```
710     //@CTK NO_OVERFLOW
```

Line 717-725 in File ITAMToken.sol

```
717     function purchaseItemOnERC20(address payable tokenAddress, uint64 appId, uint64
            itemId) external onlyNotBlackList returns(bool) {
718         uint256 itemAmount = _getItemAmount(appId, itemId, tokenAddress);
719 
720         erc20 = ERC20(tokenAddress);
721         require(erc20.transferFrom(msg.sender, inAppAddress, itemAmount), "failed
                transferFrom");
722 
723         emit PurchaseItemOnERC20(msg.sender, tokenAddress, appId, itemId, itemAmount);
724         return true;
725     }
```

The code meets the specification.

Formal Verification Request 58

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

24, Jul 2019
34.57 ms

Line 711 in File ITAMToken.sol

```
711     //@CTK NO_BUF_OVERFLOW
```

Line 717-725 in File ITAMToken.sol

```
717     function purchaseItemOnERC20(address payable tokenAddress, uint64 appId, uint64
            itemId) external onlyNotBlackList returns(bool) {
718         uint256 itemAmount = _getItemAmount(appId, itemId, tokenAddress);
719 
720         erc20 = ERC20(tokenAddress);
721         require(erc20.transferFrom(msg.sender, inAppAddress, itemAmount), "failed
                transferFrom");
722 
723         emit PurchaseItemOnERC20(msg.sender, tokenAddress, appId, itemId, itemAmount);
724         return true;
725     }
```

The code meets the specification.

Formal Verification Request 59

Method will not encounter an assertion failure.

24, Jul 2019
41.98 ms

Line 712 in File ITAMToken.sol

```
712     //@CTK NO_ASF
```

Line 717-725 in File ITAMToken.sol

```
717     function purchaseItemOnERC20(address payable tokenAddress, uint64 appId, uint64
            itemId) external onlyNotBlackList returns(bool) {
718         uint256 itemAmount = _getItemAmount(appId, itemId, tokenAddress);
719 
720         erc20 = ERC20(tokenAddress);
721         require(erc20.transferFrom(msg.sender, inAppAddress, itemAmount), "failed
                transferFrom");
722 
723         emit PurchaseItemOnERC20(msg.sender, tokenAddress, appId, itemId, itemAmount);
724         return true;
725     }
```

The code meets the specification.

Formal Verification Request 60

purchaseItemOnERC20 correctness

24, Jul 2019
4.63 ms

Line 713-716 in File ITAMToken.sol

```
713     /*@CTK "purchaseItemOnERC20 correctness"
714       @tag assume_completion
715       @post blackLists[msg.sender] == false
716      */
```

Line 717-725 in File ITAMToken.sol

```
717     function purchaseItemOnERC20(address payable tokenAddress, uint64 appId, uint64
            itemId) external onlyNotBlackList returns(bool) {
718         uint256 itemAmount = _getItemAmount(appId, itemId, tokenAddress);
719 
720         erc20 = ERC20(tokenAddress);
721         require(erc20.transferFrom(msg.sender, inAppAddress, itemAmount), "failed
                transferFrom");
722 
723         emit PurchaseItemOnERC20(msg.sender, tokenAddress, appId, itemId, itemAmount);
724         return true;
725     }
```

The code meets the specification.

Formal Verification Request 61

If method completes, integer overflow would not happen.

24, Jul 2019
275.95 ms

Line 727 in File ITAMToken.sol

```
727     //@CTK NO_OVERFLOW
```

Line 735-760 in File ITAMToken.sol

```
735     function purchaseItemOnITAM(uint64 appId, uint64 itemId) external onlyNotBlackList
            returns(bool) {
736         uint256 itemAmount = _getItemAmount(appId, itemId, address(this));
737 
738         /*@CTK "While in purchaseItemOnITAM"
739           @post discountInfos.length >= 0
740           @post !__should_return
741           */
742         while(discountInfos.length > 0) {
743             DiscountInfo memory discountInfo = discountInfos[discountInfos.length - 1];
744             if(discountInfo.startTime <= now) {
745                 if(now <= discountInfo.endTime) {
746                     itemAmount = itemAmount.sub(itemAmount.mul(discountInfo.percent).div
                        (100));
747                     break;
748                 }
749                 discountInfos.length--;
750             }
751             else {
752                 break;
753             }
754         }
755 
756         transfer(inAppAddress, itemAmount);
757 
758         emit PurchaseItemOnITAM(msg.sender, appId, itemId, itemAmount);
759         return true;
760     }
```

The code meets the specification.

Formal Verification Request 62

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

24, Jul 2019
23.77 ms

Line 728 in File ITAMToken.sol

```
728     //@CTK NO_BUF_OVERFLOW
```

Line 735-760 in File ITAMToken.sol

```
735     function purchaseItemOnITAM(uint64 appId, uint64 itemId) external onlyNotBlackList
            returns(bool) {
736         uint256 itemAmount = _getItemAmount(appId, itemId, address(this));
737 
738         /*@CTK "While in purchaseItemOnITAM"
739           @post discountInfos.length >= 0
740           @post !__should_return
741           */
742         while(discountInfos.length > 0) {
743             DiscountInfo memory discountInfo = discountInfos[discountInfos.length - 1];
744             if(discountInfo.startTime <= now) {
745                 if(now <= discountInfo.endTime) {
746                     itemAmount = itemAmount.sub(itemAmount.mul(discountInfo.percent).div
                        (100));
747                     break;
748                 }
749                 discountInfos.length--;
750             }
751             else {
752                 break;
753             }
754         }
755 
756         transfer(inAppAddress, itemAmount);
757 
758         emit PurchaseItemOnITAM(msg.sender, appId, itemId, itemAmount);
759         return true;
760     }
```

The code meets the specification.

Formal Verification Request 63

Method will not encounter an assertion failure.

24, Jul 2019
23.53 ms

Line 729 in File ITAMToken.sol

```
729     //@CTK NO_ASF
```

Line 735-760 in File ITAMToken.sol

```
735     function purchaseItemOnITAM(uint64 appId, uint64 itemId) external onlyNotBlackList
            returns(bool) {
736         uint256 itemAmount = _getItemAmount(appId, itemId, address(this));
737 
738         /*@CTK "While in purchaseItemOnITAM"
739           @post discountInfos.length >= 0
740           @post !__should_return
741           */
742         while(discountInfos.length > 0) {
743             DiscountInfo memory discountInfo = discountInfos[discountInfos.length - 1];
744             if(discountInfo.startTime <= now) {
745                 if(now <= discountInfo.endTime) {
746                     itemAmount = itemAmount.sub(itemAmount.mul(discountInfo.percent).div
                        (100));
```

```
747                 break;
748             }
749             discountInfos.length--;
750         }
751         else {
752             break;
753         }
754     }
755
756     transfer(inAppAddress, itemAmount);
757
758     emit PurchaseItemOnITAM(msg.sender, appId, itemId, itemAmount);
759     return true;
760 }
```

The code meets the specification.

Formal Verification Request 64

purchaseItemOnITAM correctness

24, Jul 2019
21.32 ms

Line 730-734 in File ITAMToken.sol

```
730 /*@CTK "purchaseItemOnITAM correctness"
731   @tag assume_completion
732   @post blackLists[msg.sender] == false
733   @post __return == true
734  */
```

Line 735-760 in File ITAMToken.sol

```
735 function purchaseItemOnITAM(uint64 appId, uint64 itemId) external onlyNotBlackList
        returns(bool) {
736     uint256 itemAmount = _getItemAmount(appId, itemId, address(this));
737
738     /*@CTK "While in purchaseItemOnITAM"
739       @post discountInfos.length >= 0
740       @post !__should_return
741       */
742     while(discountInfos.length > 0) {
743         DiscountInfo memory discountInfo = discountInfos[discountInfos.length - 1];
744         if(discountInfo.startTime <= now) {
745             if(now <= discountInfo.endTime) {
746                 itemAmount = itemAmount.sub(itemAmount.mul(discountInfo.percent).div
                        (100));
747                 break;
748             }
749             discountInfos.length--;
750         }
751         else {
752             break;
753         }
754     }
755
756     transfer(inAppAddress, itemAmount);
```

```
757
758         emit PurchaseItemOnITAM(msg.sender, appId, itemId, itemAmount);
759         return true;
760     }
```

The code meets the specification.

Formal Verification Request 65

If method completes, integer overflow would not happen.

24, Jul 2019
59.1 ms

Line 762 in File ITAMToken.sol

```
762     //@CTK NO_OVERFLOW
```

Line 771-777 in File ITAMToken.sol

```
771     function purchaseItemOnEther(uint64 appId, uint64 itemId) external payable
            onlyNotBlackList returns(bool) {
772         uint256 itemAmount = _getItemAmount(appId, itemId, address(0));
773         require(itemAmount == msg.value, "wrong quantity");
774 
775         emit PurchaseItemOnEther(msg.sender, appId, itemId, msg.value);
776         return true;
777     }
```

The code meets the specification.

Formal Verification Request 66

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

24, Jul 2019
9.93 ms

Line 763 in File ITAMToken.sol

```
763     //@CTK NO_BUF_OVERFLOW
```

Line 771-777 in File ITAMToken.sol

```
771     function purchaseItemOnEther(uint64 appId, uint64 itemId) external payable
            onlyNotBlackList returns(bool) {
772         uint256 itemAmount = _getItemAmount(appId, itemId, address(0));
773         require(itemAmount == msg.value, "wrong quantity");
774 
775         emit PurchaseItemOnEther(msg.sender, appId, itemId, msg.value);
776         return true;
777     }
```

The code meets the specification.

Formal Verification Request 67

Method will not encounter an assertion failure.

24, Jul 2019
6.47 ms

Line 764 in File ITAMToken.sol

```
764     //@CTK NO_ASF
```

Line 771-777 in File ITAMToken.sol

```
771     function purchaseItemOnEther(uint64 appId, uint64 itemId) external payable
            onlyNotBlackList returns(bool) {
772         uint256 itemAmount = _getItemAmount(appId, itemId, address(0));
773         require(itemAmount == msg.value, "wrong quantity");
774 
775         emit PurchaseItemOnEther(msg.sender, appId, itemId, msg.value);
776         return true;
777     }
```

The code meets the specification.

Formal Verification Request 68

purchaseItemOnEther correctness

24, Jul 2019
8.06 ms

Line 765-770 in File ITAMToken.sol

```
765     /*@CTK "purchaseItemOnEther correctness"
766       @tag assume_completion
767       @post blackLists[msg.sender] == false
768       @post items[appId][itemId][0x0] == msg.value
769       @post msg.value > 0
770      */
```

Line 771-777 in File ITAMToken.sol

```
771     function purchaseItemOnEther(uint64 appId, uint64 itemId) external payable
            onlyNotBlackList returns(bool) {
772         uint256 itemAmount = _getItemAmount(appId, itemId, address(0));
773         require(itemAmount == msg.value, "wrong quantity");
774 
775         emit PurchaseItemOnEther(msg.sender, appId, itemId, msg.value);
776         return true;
777     }
```

The code meets the specification.

Formal Verification Request 69

If method completes, integer overflow would not happen.

24, Jul 2019
50.9 ms

Line 780 in File ITAMToken.sol

```
780     //@CTK NO_OVERFLOW
```

Line 791-837 in File ITAMToken.sol

```
791     function resetPurchaseInAppDiscountInfo(uint[] memory startTimes, uint[] memory
            endTimes, uint8[] memory percents) public onlyGameMaster returns(bool) {
792         require(startTimes.length == endTimes.length && endTimes.length == percents.
                length);
793         discountInfos.length = 0;
794 
795         uint prevStartTime = 2 ** 256 - 1;
796         uint prevEndTime = prevStartTime;
797         uint startTime;
798         uint endTime;
799         uint8 percent;
800         /*@CTK "forLoop in resetPurchaseInAppDiscountInfo"
801           @tag assume_completion
802           @var uint prevStartTime
803           @var uint prevEndTime
804           @var uint[] startTimes
805           @var uint[] endTimes
806           @var uint8[] percents
807           @var ITAMToken this
808           @pre startTimes.length == endTimes.length
809           @pre startTimes.length == percents.length
810           @pre discountInfos.length == 0
811           @pre prevStartTime == prevEndTime
812           @inv startTimes == startTimes__pre
813           @inv endTimes == endTimes__pre
814           @inv percents == percents__pre
815           @inv i <= startTimes.length
816           @post i >= startTimes.length
817 //          @post __post.discountInfos.length == startTimes.length
818           @post !__should_return
819          */
820         for(uint8 i = 0; i < startTimes.length; i++) {
821             startTime = startTimes[i];
822             endTime = endTimes[i];
823             percent = percents[i];
824 
825             require(prevStartTime > startTime, "prevStartTime should be bigger than
                    current start time");
826             require(prevEndTime > endTime, "prevEndTime should be bigger than current
                    end time");
827             require(startTime < endTime, "endTime should be bigger than startTime");
828             require(0 < percent && percent <= 100, "invalid percent");
829 
830             discountInfos.push(DiscountInfo(startTime, endTime, percent));
831 
832             prevStartTime = startTime;
833             prevEndTime = endTime;
834         }
835 
836         return true;
837     }
```

The code meets the specification.

Formal Verification Request 70

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

24, Jul 2019
1.36 ms

Line 781 in File ITAMToken.sol

```
781     //@CTK NO_BUF_OVERFLOW
```

Line 791-837 in File ITAMToken.sol

```
791     function resetPurchaseInAppDiscountInfo(uint[] memory startTimes, uint[] memory
            endTimes, uint8[] memory percents) public onlyGameMaster returns(bool) {
792         require(startTimes.length == endTimes.length && endTimes.length == percents.
                length);
793         discountInfos.length = 0;
794 
795         uint prevStartTime = 2 ** 256 - 1;
796         uint prevEndTime = prevStartTime;
797         uint startTime;
798         uint endTime;
799         uint8 percent;
800         /*@CTK "forLoop in resetPurchaseInAppDiscountInfo"
801           @tag assume_completion
802           @var uint prevStartTime
803           @var uint prevEndTime
804           @var uint[] startTimes
805           @var uint[] endTimes
806           @var uint8[] percents
807           @var ITAMToken this
808           @pre startTimes.length == endTimes.length
809           @pre startTimes.length == percents.length
810           @pre discountInfos.length == 0
811           @pre prevStartTime == prevEndTime
812           @inv startTimes == startTimes__pre
813           @inv endTimes == endTimes__pre
814           @inv percents == percents__pre
815           @inv i <= startTimes.length
816           @post i >= startTimes.length
817 //          @post __post.discountInfos.length == startTimes.length
818           @post !__should_return
819         */
820         for(uint8 i = 0; i < startTimes.length; i++) {
821             startTime = startTimes[i];
822             endTime = endTimes[i];
823             percent = percents[i];
824 
825             require(prevStartTime > startTime, "prevStartTime should be bigger than
                    current start time");
826             require(prevEndTime > endTime, "prevEndTime should be bigger than current
                    end time");
827             require(startTime < endTime, "endTime should be bigger than startTime");
828             require(0 < percent && percent <= 100, "invalid percent");
829 
830             discountInfos.push(DiscountInfo(startTime, endTime, percent));
831 
832             prevStartTime = startTime;
```

```
833             prevEndTime = endTime;
834         }
835
836         return true;
837     }
```

The code meets the specification.

Formal Verification Request 71

Method will not encounter an assertion failure.

24, Jul 2019
0.86 ms

Line 782 in File ITAMToken.sol

```
782     //@CTK NO_ASF
```

Line 791-837 in File ITAMToken.sol

```
791     function resetPurchaseInAppDiscountInfo(uint[] memory startTimes, uint[] memory
            endTimes, uint8[] memory percents) public onlyGameMaster returns(bool) {
792         require(startTimes.length == endTimes.length && endTimes.length == percents.
                length);
793         discountInfos.length = 0;
794
795         uint prevStartTime = 2 ** 256 - 1;
796         uint prevEndTime = prevStartTime;
797         uint startTime;
798         uint endTime;
799         uint8 percent;
800         /*@CTK "forLoop in resetPurchaseInAppDiscountInfo"
801           @tag assume_completion
802           @var uint prevStartTime
803           @var uint prevEndTime
804           @var uint[] startTimes
805           @var uint[] endTimes
806           @var uint8[] percents
807           @var ITAMToken this
808           @pre startTimes.length == endTimes.length
809           @pre startTimes.length == percents.length
810           @pre discountInfos.length == 0
811           @pre prevStartTime == prevEndTime
812           @inv startTimes == startTimes__pre
813           @inv endTimes == endTimes__pre
814           @inv percents == percents__pre
815           @inv i <= startTimes.length
816           @post i >= startTimes.length
817 //          @post __post.discountInfos.length == startTimes.length
818           @post !__should_return
819          */
820         for(uint8 i = 0; i < startTimes.length; i++) {
821             startTime = startTimes[i];
822             endTime = endTimes[i];
823             percent = percents[i];
824
```

```
825             require(prevStartTime > startTime, "prevStartTime should be bigger than
                    current start time");
826             require(prevEndTime > endTime, "prevEndTime should be bigger than current
                    end time");
827             require(startTime < endTime, "endTime should be bigger than startTime");
828             require(0 < percent && percent <= 100, "invalid percent");
829
830             discountInfos.push(DiscountInfo(startTime, endTime, percent));
831
832             prevStartTime = startTime;
833             prevEndTime = endTime;
834         }
835
836         return true;
837     }
```

The code meets the specification.

Formal Verification Request 72

resetPurchaseInAppDiscountInfo correctness

24, Jul 2019
9.84 ms

Line 783-790 in File ITAMToken.sol

```
783     /*@CTK "resetPurchaseInAppDiscountInfo correctness"
784       @tag assume_completion
785       @post msg.sender == gameMaster
786       @post startTimes.length == endTimes.length
787       @post startTimes.length == percents.length
788 //      @post __post.discountInfos.length == startTimes.length
789       @post __return == true
790      */
```

Line 791-837 in File ITAMToken.sol

```
791     function resetPurchaseInAppDiscountInfo(uint[] memory startTimes, uint[] memory
            endTimes, uint8[] memory percents) public onlyGameMaster returns(bool) {
792         require(startTimes.length == endTimes.length && endTimes.length == percents.
                length);
793         discountInfos.length = 0;
794
795         uint prevStartTime = 2 ** 256 - 1;
796         uint prevEndTime = prevStartTime;
797         uint startTime;
798         uint endTime;
799         uint8 percent;
800         /*@CTK "forLoop in resetPurchaseInAppDiscountInfo"
801           @tag assume_completion
802           @var uint prevStartTime
803           @var uint prevEndTime
804           @var uint[] startTimes
805           @var uint[] endTimes
806           @var uint8[] percents
807           @var ITAMToken this
808           @pre startTimes.length == endTimes.length
```

```
809           @pre startTimes.length == percents.length
810           @pre discountInfos.length == 0
811           @pre prevStartTime == prevEndTime
812           @inv startTimes == startTimes__pre
813           @inv endTimes == endTimes__pre
814           @inv percents == percents__pre
815           @inv i <= startTimes.length
816           @post i >= startTimes.length
817   //        @post __post.discountInfos.length == startTimes.length
818           @post !__should_return
819          */
820         for(uint8 i = 0; i < startTimes.length; i++) {
821             startTime = startTimes[i];
822             endTime = endTimes[i];
823             percent = percents[i];
824
825             require(prevStartTime > startTime, "prevStartTime should be bigger than
                    current start time");
826             require(prevEndTime > endTime, "prevEndTime should be bigger than current
                    end time");
827             require(startTime < endTime, "endTime should be bigger than startTime");
828             require(0 < percent && percent <= 100, "invalid percent");
829
830             discountInfos.push(DiscountInfo(startTime, endTime, percent));
831
832             prevStartTime = startTime;
833             prevEndTime = endTime;
834         }
835
836         return true;
837     }
```

The code meets the specification.

Formal Verification Request 73

forLoop in createOrUpdateItem__Generated

24, Jul 2019
50.78 ms

(Loop) Line 678-691 in File ITAMToken.sol

```
678         /*@CTK "forLoop in createOrUpdateItem"
679           @var uint64 appId
680           @var uint64[] itemIds
681           @var address[] tokenAddresses
682           @var uint256[] values
683           @var ITAMToken this
684           @var uint itemLength
685           @pre itemLength == itemIds.length
686           @pre itemLength == tokenAddresses.length
687           @pre itemLength == values.length
688           @inv i <= itemLength
689           @post i >= itemLength
690           @post !__should_return
691           */
```

(Loop) Line 678-698 in File ITAMToken.sol

```
678         /*@CTK "forLoop in createOrUpdateItem"
679           @var uint64 appId
680           @var uint64[] itemIds
681           @var address[] tokenAddresses
682           @var uint256[] values
683           @var ITAMToken this
684           @var uint itemLength
685           @pre itemLength == itemIds.length
686           @pre itemLength == tokenAddresses.length
687           @pre itemLength == values.length
688           @inv i <= itemLength
689           @post i >= itemLength
690           @post !__should_return
691           */
692         for(uint16 i = 0; i < itemLength; i++) {
693             itemId = itemIds[i];
694             tokenAddress = tokenAddresses[i];
695             value = values[i];
696 
697             items[appId][itemId][tokenAddress] = value;
698         }
```

The code meets the specification.

Formal Verification Request 74

While in purchaseItemOnITAM__Generated

24, Jul 2019
149.19 ms

(Loop) Line 738-741 in File ITAMToken.sol

```
738         /*@CTK "While in purchaseItemOnITAM"
739           @post discountInfos.length >= 0
740           @post !__should_return
741           */
```

(Loop) Line 738-754 in File ITAMToken.sol

```
738         /*@CTK "While in purchaseItemOnITAM"
739           @post discountInfos.length >= 0
740           @post !__should_return
741           */
742         while(discountInfos.length > 0) {
743             DiscountInfo memory discountInfo = discountInfos[discountInfos.length - 1];
744             if(discountInfo.startTime <= now) {
745                 if(now <= discountInfo.endTime) {
746                     itemAmount = itemAmount.sub(itemAmount.mul(discountInfo.percent).div
                        (100));
747                     break;
748                 }
749                 discountInfos.length--;
750             }
751             else {
752                 break;
```

```
753             }
754         }
```

The code meets the specification.

Formal Verification Request 75

forLoop in resetPurchaseInAppDiscountInfo__Generated

24, Jul 2019
106.06 ms

(Loop) Line 800-819 in File ITAMToken.sol

```
800         /*@CTK "forLoop in resetPurchaseInAppDiscountInfo"
801           @tag assume_completion
802           @var uint prevStartTime
803           @var uint prevEndTime
804           @var uint[] startTimes
805           @var uint[] endTimes
806           @var uint8[] percents
807           @var ITAMToken this
808           @pre startTimes.length == endTimes.length
809           @pre startTimes.length == percents.length
810           @pre discountInfos.length == 0
811           @pre prevStartTime == prevEndTime
812           @inv startTimes == startTimes__pre
813           @inv endTimes == endTimes__pre
814           @inv percents == percents__pre
815           @inv i <= startTimes.length
816           @post i >= startTimes.length
817 //          @post __post.discountInfos.length == startTimes.length
818           @post !__should_return
819          */
```

(Loop) Line 800-834 in File ITAMToken.sol

```
800         /*@CTK "forLoop in resetPurchaseInAppDiscountInfo"
801           @tag assume_completion
802           @var uint prevStartTime
803           @var uint prevEndTime
804           @var uint[] startTimes
805           @var uint[] endTimes
806           @var uint8[] percents
807           @var ITAMToken this
808           @pre startTimes.length == endTimes.length
809           @pre startTimes.length == percents.length
810           @pre discountInfos.length == 0
811           @pre prevStartTime == prevEndTime
812           @inv startTimes == startTimes__pre
813           @inv endTimes == endTimes__pre
814           @inv percents == percents__pre
815           @inv i <= startTimes.length
816           @post i >= startTimes.length
817 //          @post __post.discountInfos.length == startTimes.length
818           @post !__should_return
819          */
820         for(uint8 i = 0; i < startTimes.length; i++) {
```

```
821             startTime = startTimes[i];
822             endTime = endTimes[i];
823             percent = percents[i];
824
825             require(prevStartTime > startTime, "prevStartTime should be bigger than
                    current start time");
826             require(prevEndTime > endTime, "prevEndTime should be bigger than current
                    end time");
827             require(startTime < endTime, "endTime should be bigger than startTime");
828             require(0 < percent && percent <= 100, "invalid percent");
829
830             discountInfos.push(DiscountInfo(startTime, endTime, percent));
831
832             prevStartTime = startTime;
833             prevEndTime = endTime;
834         }
```

The code meets the specification.

Source Code with CertiK Labels

File ITAMToken.sol

```
1  // File: openzeppelin-solidity/contracts/math/SafeMath.sol
2
3  pragma solidity ^0.5.2;
4
5  /**
6   * @title SafeMath
7   * @dev Unsigned math operations with safety checks that revert on error
8   */
9  library SafeMath {
10     /**
11      * @dev Multiplies two unsigned integers, reverts on overflow.
12      */
13     /*@CTK "SafeMath mul"
14       @post (a > 0) && (((a * b) / a) != b) -> __reverted
15       @post __reverted -> (a > 0) && (((a * b) / a) != b)
16       @post !__reverted -> __return == a * b
17       @post !__reverted == !__has_overflow
18      */
19     function mul(uint256 a, uint256 b) internal pure returns (uint256) {
20         // Gas optimization: this is cheaper than requiring 'a' not being zero, but the
21         // benefit is lost if 'b' is also tested.
22         // See: https://github.com/OpenZeppelin/openzeppelin-solidity/pull/522
23         if (a == 0) {
24             return 0;
25         }
26
27         uint256 c = a * b;
28         require(c / a == b);
29
30         return c;
31     }
32
33     /**
34      * @dev Integer division of two unsigned integers truncating the quotient, reverts
          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      */
41     function div(uint256 a, uint256 b) internal pure returns (uint256) {
42         // Solidity only automatically asserts when dividing by 0
43         require(b > 0);
44         uint256 c = a / b;
45         // assert(a == b * c + a % b); // There is no case in which this doesn't hold
46
47         return c;
48     }
49
50     /**
51      * @dev Subtracts two unsigned integers, reverts on overflow (i.e. if subtrahend
          is greater than minuend).
52      */
```

```
53      /*@CTK "SafeMath sub"
54        @post (a < b) == __reverted
55        @post !__reverted -> __return == a - b
56        @post !__reverted -> !__has_overflow
57      */
58      function sub(uint256 a, uint256 b) internal pure returns (uint256) {
59          require(b <= a);
60          uint256 c = a - b;
61
62          return c;
63      }
64
65      /**
66       * @dev Adds two unsigned integers, reverts on overflow.
67       */
68      /*@CTK "SafeMath add"
69        @post (a + b < a || a + b < b) == __reverted
70        @post !__reverted -> __return == a + b
71        @post !__reverted -> !__has_overflow
72       */
73      function add(uint256 a, uint256 b) internal pure returns (uint256) {
74          uint256 c = a + b;
75          require(c >= a);
76
77          return c;
78      }
79
80      /**
81       * @dev Divides two unsigned integers and returns the remainder (unsigned integer
             modulo),
82       * reverts when dividing by zero.
83       */
84      /*@CTK "SafeMath mod"
85        @post (b == 0) == __reverted
86        @post !__reverted -> b != 0
87        @post !__reverted -> __return == a % b
88        @post !__reverted -> !__has_overflow
89       */
90      function mod(uint256 a, uint256 b) internal pure returns (uint256) {
91          require(b != 0);
92          return a % b;
93      }
94  }
95
96  // File: openzeppelin-solidity/contracts/token/ERC20/IERC20.sol
97
98  /**
99   * @title ERC20 interface
100  * @dev see https://github.com/ethereum/EIPs/issues/20
101  */
102 interface IERC20 {
103     function transfer(address to, uint256 value) external returns (bool);
104
105     function approve(address spender, uint256 value) external returns (bool);
106
107     function transferFrom(address from, address to, uint256 value) external returns (
            bool);
108
```

```
109     function totalSupply() external view returns (uint256);
110
111     function balanceOf(address who) external view returns (uint256);
112
113     function allowance(address owner, address spender) external view returns (uint256)
            ;
114
115     event Transfer(address indexed from, address indexed to, uint256 value);
116
117     event Approval(address indexed owner, address indexed spender, uint256 value);
118 }
119
120 // File: openzeppelin-solidity/contracts/token/ERC20/ERC20.sol
121
122
123
124 /**
125  * @title Standard ERC20 token
126  *
127  * @dev Implementation of the basic standard token.
128  * https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
129  * Originally based on code by FirstBlood:
130  * https://github.com/Firstbloodio/token/blob/master/smart_contract/FirstBloodToken.
        sol
131  *
132  * This implementation emits additional Approval events, allowing applications to
        reconstruct the allowance status for
133  * all accounts just by listening to said events. Note that this isn't required by the
         specification, and other
134  * compliant implementations may not do it.
135  */
136 contract ERC20 {
137     using SafeMath for uint256;
138
139     mapping (address => uint256) private _balances;
140
141     mapping (address => mapping (address => uint256)) private _allowed;
142
143     uint256 private _totalSupply;
144
145     event Transfer(address indexed from, address indexed to, uint256 value);
146
147     event Approval(address indexed owner, address indexed spender, uint256 value);
148
149     /**
150      * @dev Total number of tokens in existence
151      */
152     /*@CTK totalSupply
153       @post __return == _totalSupply
154      */
155     function totalSupply() public view returns (uint256) {
156         return _totalSupply;
157     }
158
159     /**
160      * @dev Gets the balance of the specified address.
161      * @param owner The address to query the balance of.
162      * @return An uint256 representing the amount owned by the passed address.
```

```
163      */
164     /*@CTK balanceOf
165       @post __return == _balances[owner]
166      */
167     function balanceOf(address owner) public view returns (uint256) {
168         return _balances[owner];
169     }
170 
171     /**
172      * @dev Function to check the amount of tokens that an owner allowed to a spender.
173      * @param owner address The address which owns the funds.
174      * @param spender address The address which will spend the funds.
175      * @return A uint256 specifying the amount of tokens still available for the
             spender.
176      */
177     /*@CTK allowance
178       @post __return == _allowed[owner][spender]
179      */
180     function allowance(address owner, address spender) public view returns (uint256) {
181         return _allowed[owner][spender];
182     }
183 
184     /**
185      * @dev Transfer token for a specified address
186      * @param to The address to transfer to.
187      * @param value The amount to be transferred.
188      */
189     /*@CTK transfer
190       @tag assume_completion
191       @pre msg.sender != to
192       @post to != address(0)
193       @post value <= _balances[msg.sender]
194       @post __post._balances[to] == _balances[to] + value
195       @post __post._balances[msg.sender] == _balances[msg.sender] - value
196      */
197     function transfer(address to, uint256 value) public returns (bool) {
198         _transfer(msg.sender, to, value);
199         return true;
200     }
201 
202     /**
203      * @dev Approve the passed address to spend the specified amount of tokens on
             behalf of msg.sender.
204      * Beware that changing an allowance with this method brings the risk that someone
              may use both the old
205      * and the new allowance by unfortunate transaction ordering. One possible
             solution to mitigate this
206      * race condition is to first reduce the spender's allowance to 0 and set the
             desired value afterwards:
207      * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
208      * @param spender The address which will spend the funds.
209      * @param value The amount of tokens to be spent.
210      */
211     /*@CTK approve
212       @tag assume_completion
213       @post spender != address(0)
214       @post __post._allowed[msg.sender][spender] == value
215      */
```

```
216     function approve(address spender, uint256 value) public returns (bool) {
217         _approve(msg.sender, spender, value);
218         return true;
219     }
220 
221     /**
222      * @dev Transfer tokens from one address to another.
223      * Note that while this function emits an Approval event, this is not required as
             per the specification,
224      * and other compliant implementations may not emit the event.
225      * @param from address The address which you want to send tokens from
226      * @param to address The address which you want to transfer to
227      * @param value uint256 the amount of tokens to be transferred
228      */
229     /*@CTK transfer_from
230       @tag assume_completion
231       @pre from != to
232       @post to != address(0)
233       @post value <= _allowed[from][msg.sender]
234       @post __post._balances[from] == _balances[from] - value
235       @post __post._balances[to] == _balances[to] + value
236       @post __post._allowed[from][msg.sender] ==
237       _allowed[from][msg.sender] - value
238      */
239     function transferFrom(address from, address to, uint256 value) public returns (
            bool) {
240         _transfer(from, to, value);
241         _approve(from, msg.sender, _allowed[from][msg.sender].sub(value));
242         return true;
243     }
244 
245     /**
246      * @dev Transfer token for a specified addresses
247      * @param from The address to transfer from.
248      * @param to The address to transfer to.
249      * @param value The amount to be transferred.
250      */
251     function _transfer(address from, address to, uint256 value) internal {
252         require(to != address(0));
253 
254         _balances[from] = _balances[from].sub(value);
255         _balances[to] = _balances[to].add(value);
256         emit Transfer(from, to, value);
257     }
258 
259     /**
260      * @dev Internal function that mints an amount of the token and assigns it to
261      * an account. This encapsulates the modification of balances such that the
262      * proper events are emitted.
263      * @param account The account that will receive the created tokens.
264      * @param value The amount that will be created.
265      */
266     /*@CTK mint
267       @tag assume_completion
268       @post account != 0
269       @post __post._totalSupply == _totalSupply + value
270       @post __post._balances[account] == _balances[account] + value
271      */
```

```
272     function mint(address account, uint256 value) internal {
273         require(account != address(0));
274 
275         _totalSupply = _totalSupply.add(value);
276         _balances[account] = _balances[account].add(value);
277         emit Transfer(address(0), account, value);
278     }
279 
280     /**
281      * @dev Approve an address to spend another addresses' tokens.
282      * @param owner The address that owns the tokens.
283      * @param spender The address that will spend the tokens.
284      * @param value The number of tokens that can be spent.
285      */
286     function _approve(address owner, address spender, uint256 value) internal {
287         require(spender != address(0));
288         require(owner != address(0));
289 
290         _allowed[owner][spender] = value;
291         emit Approval(owner, spender, value);
292     }
293 
294     function _burn(address account, uint256 value) internal {
295         require(account != address(0));
296 
297         _totalSupply = _totalSupply.sub(value);
298         _balances[account] = _balances[account].sub(value);
299         emit Transfer(account, address(0), value);
300     }
301 }
302 
303 // File: openzeppelin-solidity/contracts/token/ERC20/ERC20Capped.sol
304 
305 
306 /**
307  * @title Capped token
308  * @dev Mintable token with a token cap.
309  */
310 contract ERC20Capped is ERC20 {
311     uint256 private _cap;
312 
313     /*@CTK ERC20Capped
314       @tag assume_completion
315       @post cap > 0
316       @post __post._cap == cap
317      */
318     constructor (uint256 cap) public {
319         require(cap > 0);
320         _cap = cap;
321     }
322 
323     /**
324      * @return the cap for the token minting.
325      */
326     /*@CTK cap
327       @post __return == _cap
328      */
329     function cap() public view returns (uint256) {
```

```
330         return _cap;
331     }
332 
333     /*@CTK _mint
334       @tag assume_completion
335       @post _totalSupply + value <= _cap
336       @post account != address(0)
337       @post __post._totalSupply == _totalSupply + value
338       @post __post._balances[account] == _balances[account] + value
339      */
340     function _mint(address account, uint256 value) internal {
341         require(totalSupply().add(value) <= _cap);
342         super.mint(account, value);
343     }
344 }
345 
346 // File: contracts/ITAMToken.sol
347 
348 contract ITAMToken is ERC20Capped {
349     string public name = "ITAM";
350     string public symbol = "ITAM";
351     uint8 public decimals = 18;
352     uint256 constant TOTAL_CAP = 2500000000 ether;
353 
354     address public owner;
355     address public gameMaster;
356     mapping(address => bool) public blackLists;
357 
358     struct DiscountInfo {
359         uint startTime;
360         uint endTime;
361         uint8 percent;
362     }
363 
364     DiscountInfo[] public discountInfos;
365 
366     uint8 public unlockCount = 0;
367     address public strategicSaleAddress;
368     uint[] public strategicSaleReleaseCaps = [15000000 ether, 15000000 ether, 15000000
          ether,
369                                               15000000 ether, 15000000 ether, 15000000
                                                  ether,
370                                               15000000 ether, 22500000 ether, 22500000
                                                  ether];
371 
372     address public privateSaleAddress;
373     uint[] public privateSaleReleaseCaps = [97500000 ether, 97500000 ether, 97500000
         ether,
374                                             97500000 ether, 130000000 ether, 130000000
                                                ether];
375 
376     address public publicSaleAddress;
377     uint public publicSaleReleaseCap = 200000000 ether;
378 
379     address public teamAddress;
380     uint[] public teamReleaseCaps = [0, 0, 0, 0, 0, 0,
381                                      12500000 ether, 12500000 ether, 12500000 ether,
382                                      12500000 ether, 12500000 ether, 12500000 ether,
```

```
                                    12500000 ether, 12500000 ether, 12500000 ether,
                                    12500000 ether, 12500000 ether, 12500000 ether,
                                    12500000 ether, 12500000 ether, 12500000 ether,
                                    12500000 ether, 12500000 ether, 12500000 ether,
                                    12500000 ether, 12500000 ether];

    address public advisorAddress;
    uint[] public advisorReleaseCaps = [0, 0, 0, 25000000 ether, 0, 25000000 ether,
                                        0, 25000000 ether, 0, 25000000 ether, 0, 25000000
                                            ether];

    address public marketingAddress;
    uint[] public marketingReleaseCaps = [100000000 ether, 25000000 ether, 25000000
        ether,
                                          25000000 ether, 25000000 ether, 25000000 ether,
                                          25000000 ether, 25000000 ether, 25000000 ether,
                                          25000000 ether, 25000000 ether, 25000000 ether];

    address public ecoAddress;
    uint[] public ecoReleaseCaps = [50000000 ether, 50000000 ether, 50000000 ether,
                                    50000000 ether, 50000000 ether, 50000000 ether,
                                    50000000 ether, 50000000 ether, 50000000 ether,
                                    50000000 ether, 50000000 ether, 50000000 ether,
                                    50000000 ether, 50000000 ether, 50000000 ether];
    address payable public inAppAddress;

    ERC20 erc20;

    // appId => itemId => tokenAddress => amount
    mapping(uint64 => mapping(uint64 => mapping(address => uint256))) items;

    event Unlock(uint8 unlockCount);
    event WithdrawEther(address indexed _to, uint256 amount);
    event PurchaseItemOnEther(address indexed _spender, uint64 appId, uint64 itemId,
        uint256 amount);
    event PurchaseItemOnITAM(address indexed _spender, uint64 appId, uint64 itemId,
        uint256 amount);
    event PurchaseItemOnERC20(address indexed _spender, address indexed _tokenAddress,
         uint64 appId, uint64 itemId, uint256 amount);
    event SetItem(uint64 appId);

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK "ITAMToken constructor"
      @tag assume_completion
      @post __post.owner == _owner
      @post __post.gameMaster == _gameMaster
      @post __post.strategicSaleAddress == _strategicSaleAddress
      @post __post.privateSaleAddress == _privateSaleAddress
      @post __post.publicSaleAddress == _publicSaleAddress
      @post __post.teamAddress == _teamAddress
      @post __post.advisorAddress == _advisorAddress
      @post __post.marketingAddress == _marketingAddress
      @post __post.ecoAddress == _ecoAddress
      @post __post.inAppAddress == _inAppAddress
     */
```

```
435     constructor(address _owner, address _gameMaster, address _strategicSaleAddress,
            address _privateSaleAddress, address _publicSaleAddress, address _teamAddress,
             address _advisorAddress, address _marketingAddress, address _ecoAddress,
436                 address payable _inAppAddress) public ERC20Capped(TOTAL_CAP) {
437         owner = _owner;
438         gameMaster = _gameMaster;
439         strategicSaleAddress = _strategicSaleAddress;
440         privateSaleAddress = _privateSaleAddress;
441         publicSaleAddress = _publicSaleAddress;
442         teamAddress = _teamAddress;
443         advisorAddress = _advisorAddress;
444         marketingAddress = _marketingAddress;
445         ecoAddress = _ecoAddress;
446         inAppAddress = _inAppAddress;
447     }
448
449     modifier onlyOwner {
450         require(msg.sender == owner);
451         _;
452     }
453
454     modifier onlyGameMaster {
455         require(msg.sender == gameMaster);
456         _;
457     }
458
459     //@CTK NO_OVERFLOW
460     //@CTK NO_BUF_OVERFLOW
461     //@CTK NO_ASF
462     /*@CTK setGameMaster
463       @tag assume_completion
464       @post __post.gameMaster == _gameMaster
465      */
466     function setGameMaster(address _gameMaster) public onlyOwner {
467         gameMaster = _gameMaster;
468     }
469
470     //@CTK NO_OVERFLOW
471     //@CTK NO_BUF_OVERFLOW
472     //@CTK NO_ASF
473     /*@CTK "transfer correctness"
474       @tag assume_completion
475       @post blackLists[msg.sender] == false
476       @post _to != 0x0
477       @post _value <= _balances[msg.sender]
478       @post _to != msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
              - _value
479       @post _to != msg.sender -> __post._balances[_to] == _balances[_to] + _value
480       @post _to == msg.sender -> __post._balances[msg.sender] == _balances[msg.sender]
481      */
482     function transfer(address _to, uint256 _value) public onlyNotBlackList returns (
            bool) {
483         return super.transfer(_to, _value);
484     }
485
486     //@CTK NO_OVERFLOW
487     //@CTK NO_BUF_OVERFLOW
488     //@CTK NO_ASF
```

```
489     /*@CTK "transferFrom correctness"
490       @tag assume_completion
491       @post blackLists[msg.sender] == false
492       @post _to != 0x0
493       @post _value <= _balances[_from] && _value <= _allowed[_from][msg.sender]
494       @post _to != _from -> __post._balances[_from] == _balances[_from] - _value
495       @post _to != _from -> __post._balances[_to] == _balances[_to] + _value
496       @post _to == _from -> __post._balances[_from] == _balances[_from]
497       @post __post._allowed[_from][msg.sender] == _allowed[_from][msg.sender] - _value
498      */
499     function transferFrom(address _from, address _to, uint256 _value) public
            onlyNotBlackList returns (bool) {
500         return super.transferFrom(_from, _to, _value);
501     }
502
503     //@CTK NO_OVERFLOW
504     //@CTK NO_BUF_OVERFLOW
505     //@CTK NO_ASF
506     /*@CTK "approve correctness"
507       @tag assume_completion
508       @post blackLists[msg.sender] == false
509       @post spender != address(0)
510       @post __post._allowed[msg.sender][spender] == value
511      */
512     function approve(address spender, uint256 value) public onlyNotBlackList returns (
            bool) {
513         return super.approve(spender, value);
514     }
515
516     //@CTK NO_BUF_OVERFLOW
517     //@CTK NO_ASF
518     /*@CTK "burn correctness"
519       @tag assume_completion
520       @post msg.sender == owner
521       @post __post._totalSupply == _totalSupply - value
522       @post __post._balances[msg.sender] == _balances[msg.sender] - value
523      */
524     function burn(uint256 value) public onlyOwner {
525         super._burn(msg.sender, value);
526     }
527
528     //CTK FAIL NO_OVERFLOW
529     //@CTK NO_BUF_OVERFLOW
530     //@CTK NO_ASF
531     /*CTK "unlock when unlockCount = 0"
532       @tag assume_completion
533       @pre strategicSaleReleaseCaps.length == 9
534       @pre privateSaleReleaseCaps.length == 6
535       @pre teamReleaseCaps.length == 26
536       @pre advisorReleaseCaps.length == 12
537       @pre marketingReleaseCaps.length == 12
538       @pre ecoReleaseCaps.length == 15
539       @pre unlockCount == 0
540       @post __post.unlockCount == 1
541       @post msg.sender == owner
542       @post __post._totalSupply <= _cap
543       @post __post._totalSupply == _totalSupply
544                                 + strategicSaleReleaseCaps[unlockCount]
```

```
                                + privateSaleReleaseCaps[unlockCount]
                                + publicSaleReleaseCaps
                                + teamReleaseCaps[unlockCount]
                                + advisorReleaseCaps[unlockCount]
                                + marketingReleaseCaps[unlockCount]
                                + ecoReleaseCaps[unlockCount]
    @post __post._balances[strategicSaleAddress] >= _balances[strategicSaleAddress]
        + strategicSaleReleaseCaps[unlockCount]
     */
    function unlock() public onlyOwner returns (bool) {
        uint8 _unlockCount = unlockCount;

        if(strategicSaleReleaseCaps.length > _unlockCount) {
            super._mint(strategicSaleAddress, strategicSaleReleaseCaps[_unlockCount]);
        }

        if(privateSaleReleaseCaps.length > _unlockCount) {
            super._mint(privateSaleAddress, privateSaleReleaseCaps[_unlockCount]);
        }

        if(_unlockCount == 0) {
            super._mint(publicSaleAddress, publicSaleReleaseCap);
        }

        if(teamReleaseCaps.length > _unlockCount) {
            super._mint(teamAddress, teamReleaseCaps[_unlockCount]);
        }

        if(advisorReleaseCaps.length > _unlockCount) {
            super._mint(advisorAddress, advisorReleaseCaps[_unlockCount]);
        }

        if(marketingReleaseCaps.length > _unlockCount) {
            super._mint(marketingAddress, marketingReleaseCaps[_unlockCount]);
        }

        if(ecoReleaseCaps.length > _unlockCount) {
            super._mint(ecoAddress, ecoReleaseCaps[_unlockCount]);
        }

        unlockCount++;
        return true;
    }

    //@CTK NO_OVERFLOW
    //@CTK NO_BUF_OVERFLOW
    //@CTK NO_ASF
    /*@CTK setAddresses
      @tag assume_completion
      @post msg.sender == owner
      @post __post.strategicSaleAddress == _strategicSaleAddress
      @post __post.privateSaleAddress == _privateSaleAddress
      @post __post.publicSaleAddress == _publicSaleAddress
      @post __post.teamAddress == _teamAddress
      @post __post.advisorAddress == _advisorAddress
      @post __post.marketingAddress == _marketingAddress
      @post __post.ecoAddress == _ecoAddress
      @post __post.inAppAddress == _inAppAddress
```

```
602      */
603     function setAddresses(address _strategicSaleAddress, address _privateSaleAddress,
            address _publicSaleAddress, address _teamAddress, address _advisorAddress,
            address _marketingAddress, address _ecoAddress,
604                         address payable _inAppAddress) public onlyOwner {
605         strategicSaleAddress = _strategicSaleAddress;
606         privateSaleAddress = _privateSaleAddress;
607         publicSaleAddress = _publicSaleAddress;
608         teamAddress = _teamAddress;
609         advisorAddress = _advisorAddress;
610         marketingAddress = _marketingAddress;
611         ecoAddress = _ecoAddress;
612         inAppAddress = _inAppAddress;
613     }
614
615     //@CTK NO_OVERFLOW
616     //@CTK NO_BUF_OVERFLOW
617     //@CTK NO_ASF
618     /*@CTK addToBlackList
619       @tag assume_completion
620       @post msg.sender == owner
621       @post blackLists[_to] == false
622       @post __post.blackLists[_to] == true
623      */
624     function addToBlackList(address _to) public onlyOwner {
625         require(!blackLists[_to], "already blacklist");
626         blackLists[_to] = true;
627     }
628
629     //@CTK NO_OVERFLOW
630     //@CTK NO_BUF_OVERFLOW
631     //@CTK NO_ASF
632     /*@CTK removeFromBlackList
633       @tag assume_completion
634       @post msg.sender == owner
635       @post blackLists[_to] == true
636       @post __post.blackLists[_to] == false
637      */
638     function removeFromBlackList(address _to) public onlyOwner {
639         require(blackLists[_to], "cannot found this address from blacklist");
640         blackLists[_to] = false;
641     }
642
643     modifier onlyNotBlackList {
644         require(!blackLists[msg.sender], "sender cannot call this contract");
645         _;
646     }
647
648     // can accept ether
649     function() payable external {
650
651     }
652
653     //@CTK NO_OVERFLOW
654     //@CTK NO_BUF_OVERFLOW
655     /*@CTK "withdrawEther correctness"
656       @tag assume_completion
657       @post msg.sender == owner
```

```
658      */
659     function withdrawEther(uint256 amount) public onlyOwner {
660         inAppAddress.transfer(amount);
661         emit WithdrawEther(inAppAddress, amount);
662     }
663
664     /*@CTK "createOrUpdateItem correctness"
665       @tag assume_completion
666       @post msg.sender == gameMaster
667       @post itemIds.length == tokenAddresses.length
668       @post tokenAddresses.length == values.length
669       @post __return == true
670      */
671     function createOrUpdateItem(uint64 appId, uint64[] memory itemIds, address[]
          memory tokenAddresses, uint256[] memory values) public onlyGameMaster returns(
          bool) {
672         uint itemLength = itemIds.length;
673         require(itemLength == tokenAddresses.length && tokenAddresses.length == values.
              length);
674
675         uint64 itemId;
676         address tokenAddress;
677         uint256 value;
678         /*@CTK "forLoop in createOrUpdateItem"
679           @var uint64 appId
680           @var uint64[] itemIds
681           @var address[] tokenAddresses
682           @var uint256[] values
683           @var ITAMToken this
684           @var uint itemLength
685           @pre itemLength == itemIds.length
686           @pre itemLength == tokenAddresses.length
687           @pre itemLength == values.length
688           @inv i <= itemLength
689           @post i >= itemLength
690           @post !__should_return
691           */
692         for(uint16 i = 0; i < itemLength; i++) {
693             itemId = itemIds[i];
694             tokenAddress = tokenAddresses[i];
695             value = values[i];
696
697             items[appId][itemId][tokenAddress] = value;
698         }
699
700         emit SetItem(appId);
701         return true;
702     }
703
704     function _getItemAmount(uint64 appId, uint64 itemId, address tokenAddress) private
           view returns(uint256) {
705         uint256 itemAmount = items[appId][itemId][tokenAddress];
706         require(itemAmount > 0, "invalid item id");
707         return itemAmount;
708     }
709
710     //@CTK NO_OVERFLOW
711     //@CTK NO_BUF_OVERFLOW
```

```
712     //@CTK NO_ASF
713     /*@CTK "purchaseItemOnERC20 correctness"
714       @tag assume_completion
715       @post blackLists[msg.sender] == false
716      */
717     function purchaseItemOnERC20(address payable tokenAddress, uint64 appId, uint64
            itemId) external onlyNotBlackList returns(bool) {
718         uint256 itemAmount = _getItemAmount(appId, itemId, tokenAddress);
719 
720         erc20 = ERC20(tokenAddress);
721         require(erc20.transferFrom(msg.sender, inAppAddress, itemAmount), "failed
                transferFrom");
722 
723         emit PurchaseItemOnERC20(msg.sender, tokenAddress, appId, itemId, itemAmount);
724         return true;
725     }
726 
727     //@CTK NO_OVERFLOW
728     //@CTK NO_BUF_OVERFLOW
729     //@CTK NO_ASF
730     /*@CTK "purchaseItemOnITAM correctness"
731       @tag assume_completion
732       @post blackLists[msg.sender] == false
733       @post __return == true
734      */
735     function purchaseItemOnITAM(uint64 appId, uint64 itemId) external onlyNotBlackList
             returns(bool) {
736         uint256 itemAmount = _getItemAmount(appId, itemId, address(this));
737 
738         /*@CTK "While in purchaseItemOnITAM"
739           @post discountInfos.length >= 0
740           @post !__should_return
741           */
742         while(discountInfos.length > 0) {
743             DiscountInfo memory discountInfo = discountInfos[discountInfos.length - 1];
744             if(discountInfo.startTime <= now) {
745                 if(now <= discountInfo.endTime) {
746                     itemAmount = itemAmount.sub(itemAmount.mul(discountInfo.percent).div
                            (100));
747                     break;
748                 }
749                 discountInfos.length--;
750             }
751             else {
752                 break;
753             }
754         }
755 
756         transfer(inAppAddress, itemAmount);
757 
758         emit PurchaseItemOnITAM(msg.sender, appId, itemId, itemAmount);
759         return true;
760     }
761 
762     //@CTK NO_OVERFLOW
763     //@CTK NO_BUF_OVERFLOW
764     //@CTK NO_ASF
765     /*@CTK "purchaseItemOnEther correctness"
```

```
766       @tag assume_completion
767       @post blackLists[msg.sender] == false
768       @post items[appId][itemId][0x0] == msg.value
769       @post msg.value > 0
770      */
771     function purchaseItemOnEther(uint64 appId, uint64 itemId) external payable
              onlyNotBlackList returns(bool) {
772         uint256 itemAmount = _getItemAmount(appId, itemId, address(0));
773         require(itemAmount == msg.value, "wrong quantity");
774 
775         emit PurchaseItemOnEther(msg.sender, appId, itemId, msg.value);
776         return true;
777     }
778 
779     // startTimes, endTimes should be in slow order
780     //@CTK NO_OVERFLOW
781     //@CTK NO_BUF_OVERFLOW
782     //@CTK NO_ASF
783     /*@CTK "resetPurchaseInAppDiscountInfo correctness"
784       @tag assume_completion
785       @post msg.sender == gameMaster
786       @post startTimes.length == endTimes.length
787       @post startTimes.length == percents.length
788 //      @post __post.discountInfos.length == startTimes.length
789       @post __return == true
790      */
791     function resetPurchaseInAppDiscountInfo(uint[] memory startTimes, uint[] memory
              endTimes, uint8[] memory percents) public onlyGameMaster returns(bool) {
792         require(startTimes.length == endTimes.length && endTimes.length == percents.
                  length);
793         discountInfos.length = 0;
794 
795         uint prevStartTime = 2 ** 256 - 1;
796         uint prevEndTime = prevStartTime;
797         uint startTime;
798         uint endTime;
799         uint8 percent;
800         /*@CTK "forLoop in resetPurchaseInAppDiscountInfo"
801           @tag assume_completion
802           @var uint prevStartTime
803           @var uint prevEndTime
804           @var uint[] startTimes
805           @var uint[] endTimes
806           @var uint8[] percents
807           @var ITAMToken this
808           @pre startTimes.length == endTimes.length
809           @pre startTimes.length == percents.length
810           @pre discountInfos.length == 0
811           @pre prevStartTime == prevEndTime
812           @inv startTimes == startTimes__pre
813           @inv endTimes == endTimes__pre
814           @inv percents == percents__pre
815           @inv i <= startTimes.length
816           @post i >= startTimes.length
817 //          @post __post.discountInfos.length == startTimes.length
818           @post !__should_return
819          */
820         for(uint8 i = 0; i < startTimes.length; i++) {
```

```
821             startTime = startTimes[i];
822             endTime = endTimes[i];
823             percent = percents[i];
824 
825             require(prevStartTime > startTime, "prevStartTime should be bigger than
                    current start time");
826             require(prevEndTime > endTime, "prevEndTime should be bigger than current
                    end time");
827             require(startTime < endTime, "endTime should be bigger than startTime");
828             require(0 < percent && percent <= 100, "invalid percent");
829 
830             discountInfos.push(DiscountInfo(startTime, endTime, percent));
831 
832             prevStartTime = startTime;
833             prevEndTime = endTime;
834         }
835 
836         return true;
837     }
838 }
```