**Testarea unei retele Blockchain**

Pentru inceput am decis sa testam smartcontract-ul ce implementeaza standard-ul ERC20 pentru tokens pe reteaua ETHEREUM. Am ales sa utilizam testarea functionala pentru acest contract, abordand tehnica *Partitionarii pe categorii.*

**Testam constructorul contractului**

Analizand constructorul contractului, putem observa ca acesta primeste un singur parametru, si anume numarul total de tokens existenti de tipul definit. Prin urmare, categoriile in care se imparte input-ul constructorului sunt urmatoarele:

**tokens:**

* < 0
* 0
* > 0 si < 2256
* >= 2256

**name**:

* lungime 0 sau >50
* lungime >0 si <=50

**symbol:**

* lungime 0 sau >5
* lungime >0 si <=5

Asadar, testele ce reies din aceasta partitionare sunt urmatoarele:

|  |  |  |  |
| --- | --- | --- | --- |
| **Valori de input** | | | **Output asteptat** |
| **name** | **symbol** | **tokens** |  |
| TestToken | TT | -1 | Eroare (out-of-bounds, value=-1) |
| TestToken | TT | 0 | Revert (Initial supply should be a positive integer) |
| TestToken | TT | 1.000.000 | OK |
| TestToken | TT | 2256 | Eroare (overflow) |
| string vid | TT | 1.000.000 | Revert (Name should not be empty) |
| TestToken | string vid | 1.000.000 | Revert (Symbol should not be empty) |

**Testam functia approve**

Aceasta functie primeste doi parametri: un numar de tokens si o adresa de cont EOA/SC; ea are drept efect aprobarea unei sume de tokens pe care contul de la adresa data ca parametru o poate folosi din balanta celui ce apeleaza functia si returneaza true daca s-a executat cu succes. Impartirea input-urilor pe categorii este urmatoarea:

**tokens:**

* < 0
* >= 0 si < 2256
* >= 256

**spender:**

* adresa in format invalid
* adresa in format valid

Testele construite pe baza acestor categorii sunt urmatoarele:

|  |  |  |
| --- | --- | --- |
| **Valori pentru input** | | **Output asteptat** |
| **spender** | **tokens** | **-------------------------------** |
| 0x11...1 (10 de 1; invalid) | 10 | Eroare (invalid address) |
| 0x70...8 (valid) | -1 | Eroare (out-of-bounds, value=-1) |
| 0x70...8 (valid) | 100 | OK (true) |
| 0x70...8 (valid) | 2256 | Eroare (overflow) |

**Testam functia transfer**

Functia transfer are drept parametri adresa celui catre care se trimit tokens si numarul de astfel de tokens trimisi. Functia verifica mai intai daca cel ce o apeleaza are disponibila suma de tokens in cont (in mapping-ul balances din smartcontract) pentru ca mai apoi sa modifice efectiv mapping-ul balances pentru a ilustra operatia de transfer. Aceasta intoarce true daca transferul este efectuat cu succes. Categoriile in care se impart input-urile sunt urmatoarele:

**receiver:**

* adresa in format invalid
* adresa in format valid

**tokens:**

* < 0
* >= 0 si <= balanta apelantului
* > balanta apelantului si < 2256
* >= 2256

In urma categoriilor anterior definite am construit urmatoarele teste:

|  |  |  |
| --- | --- | --- |
| **Valori pentru input** | | **Output asteptat** |
| **receiver** | **tokens** | **-------------------------------** |
| 0x11...1 (10 de 1; invalid) | 10 | Eroare (invalid address) |
| 0x70...8 (valid) | -1 | Eroare (out-of-bounds, value=-1) |
| 0x70...8 (valid) | 100 | OK (true) |
| 0x70...8 (valid) | 1.000.001 | Revert (Insufficient funds) |
| 0x70...8 (valid) | 2256 | Eroare (overflow) |

**Testam functia balanceOf**

Functia balanceOf este un simplu getter. Singurul ei scop este de a intoarce valoarea (uint) retinuta in mapping-ul balances pentru o anumita adresa primita ca parametru (tokenOwner). Input-ul se poate clasifica in doua categorii:

**tokenOwner:**

* adresa in format invalid
* adresa in format valid, dar care nu exista in mapping
* adresa in format valid, care exista in mapping

Testele sunt urmatoarele:

|  |  |
| --- | --- |
| **Valori de input (tokenOwner)** | **Output asteptat** |
| 0x11...1 (10 de 1; invalid) | Eroare (invalid address) |
| 0x3C...C (valid; nu este in mapping) | 0 |
| 0xf3...6 (valid; in mapping; owner) | 1.000.000 |

**Testam functia allowance**

Si aceasta functie, la fel ca balanceOf, este un getter. Aceasta ne permite sa interogam suma de tokens pe care contul de la adresa furnizata prin parametrul spender are dreptul sa o cheltuiasca din contul de la adresa lui tokenOwner. Categoriile definite de input-uri sunt urmatoarele:

**tokenOwner:**

* adresa in format invalid
* adresa in format valid, dar care nu exista in mapping-ul exterior
* adresa in format valid, care exista in mapping-ul exterior

**spender:**

* adresa in format invalid
* adresa in format valid, dar care nu exista in mapping-ul interior
* adresa in format valid, care exista in mapping-ul interior

Tabelul cu testele definite este urmatorul:

|  |  |  |
| --- | --- | --- |
| **Valori pentru input** | | **Output asteptat** |
| **tokenOwner** | **spender** | **-------------------------------** |
| 0x11...1 (10 de 1; invalid) | 0x70...8 (valid) | Eroare (invalid address) |
| 0xf3...6 (valid) | 0x11...1 (10 de 1; invalid) | Eroare (invalid address) |
| 0x3C...C (valid; not in outer mapping) | 0x70...8 (valid) | 0 |
| 0xf3...6 (valid; in outer mapping) | 0x3C...C (valid; not in inner mapping) | 0 |
| 0xf3...6 (valid; in outer mapping) | 0x70...8 (valid; in inner mapping) | 1000 |

**Testam functia transferFrom**

Aceasta este cea mai complexa functie din cadrul smartcontractului pe care il testam. Aceasta implementeaza functionalitatea de transfer de tokens din alt cont pentru care ai aprobare. Functia verifica mai intai faptul ca contul de la care se doreste sa se transfere (from) are suficiente fonduri, dupa care daca cel ce apeleaza functia are aprobare sa cheltuiasca suma respectiva (tokens) din contul respectiv. Dupa aceste verificari, functia modifica mapping-urile balances si spendlimit pentru a ilustra transferul de tokens de la from catre to si returneaza true la executare cu succes. Categoriile distinse pentru input-uri sunt urmatoarele:

**from:**

* adresa in format invalid
* adresa in format valid

**to:**

* adresa in format invalid
* adresa in format valid

**tokens:**

* < 0
* >= 0 si <= min(valoarea aprobata de catre from, suma detinuta de from)
* > min(valoarea aprobata de catre from, suma detinuta de from) si < 2256
* >= 2256

Tabelul cu testele scrise este cel de mai jos:

|  |  |  |  |
| --- | --- | --- | --- |
| **Valori de input** | | | **Output asteptat** |
| **from** | **to** | **tokens** | **-------------------------** |
| 0x11..1 (10 de 1; invalid) | 0x3C...C (valid) | 100 | Eroare (invalid address) |
| 0xf3...6 (valid) | 0x11..1 (10 de 1; invalid) | 100 | Eroare (invalid address) |
| 0xf3...6 (valid) | 0x3C...C (valid) | -1 | Eroare (out-of-bounds, value=-1) |
| 0xf3...6 (valid) | 0x3C...C (valid) | 100 | OK (true) |
| 0xf3...6 (valid) | 0x3C...C (valid) | 101 | Revert (Insufficient allowance) |
| 0xf3...6 (valid) | 0x3C...C (valid) | 1.000.001 | Revert (Insufficient funds) |
| 0xf3...6 (valid) | 0x3C...C (valid) | 2256 | Eroare (overflow) |

In toate apelurile din tabelul anterior am considerat drept apelant (msg.sender) contul de la adresa: **0x70997970C51812dc3A010C7d01b50e0d17dc79C8**