

```

1 // SPDX-License-Identifier: AGPL-3.0-or-later
2 pragma solidity 0.7.5;
3
4 interface IERC20 {
5     function decimals() external view returns (uint
6         8);
7     /**
8      * @dev Returns the amount of tokens in existenc
9         e.
10      */
11     function totalSupply() external view returns (uin
12         t256);
13     /**
14      * @dev Returns the amount of tokens owned by `ac
15         count`.
16      */
17     function balanceOf(address account) external view
18         returns (uint256);
19     /**
20      * @dev Moves `amount` tokens from the caller's a
21         ccount to `recipient`.
22      *
23      * Returns a boolean value indicating whether the
24         operation succeeded.
25      *
26      * Emits a {Transfer} event.
27      */
28     function transfer(address recipient, uint256 amou
29         nt) external returns (bool);
30     /**
31      * @dev Returns the remaining number of tokens th
32         at `spender` will be
33      * allowed to spend on behalf of `owner` through
34         {transferFrom}. This is
35      * zero by default.
36      *
37      * This value changes when {approve} or {transfer
38         From} are called.
39      */
40     function allowance(address owner, address spende
41         r) external view returns (uint256);
42     /**
43      * @dev Sets `amount` as the allowance of `spende
44         r` over the caller's tokens.
45      *
46      * Returns a boolean value indicating whether the
47         operation succeeded.
48      *
49      * IMPORTANT: Beware that changing an allowance w
50         ith this method brings the risk
51      * that someone may use both the old and the new
52         allowance by unfortunate
53      * transaction ordering. One possible solution to
54         mitigate this race
55      * condition is to first reduce the spender's all
56         owance to 0 and set the
57      * desired value afterwards:

```

```

1 // SPDX-License-Identifier: AGPL-3.0-or-later
2 pragma solidity 0.7.5;
3 // Only change to generate diff
4
5 interface IERC20 {
6     function decimals() external view returns (uint
7         8);
8     /**
9      * @dev Returns the amount of tokens in existenc
10         e.
11      */
12     function totalSupply() external view returns (uin
13         t256);
14     /**
15      * @dev Returns the amount of tokens owned by `ac
16         count`.
17      */
18     function balanceOf(address account) external view
19         returns (uint256);
20     /**
21      * @dev Moves `amount` tokens from the caller's a
22         ccount to `recipient`.
23      *
24      * Returns a boolean value indicating whether the
25         operation succeeded.
26      *
27      * Emits a {Transfer} event.
28      */
29     function transfer(address recipient, uint256 amou
30         nt) external returns (bool);
31     /**
32      * @dev Returns the remaining number of tokens th
33         at `spender` will be
34      * allowed to spend on behalf of `owner` through
35         {transferFrom}. This is
36      * zero by default.
37      *
38      * This value changes when {approve} or {transfer
39         From} are called.
40      */
41     function allowance(address owner, address spende
42         r) external view returns (uint256);
43     /**
44      * @dev Sets `amount` as the allowance of `spende
45         r` over the caller's tokens.
46      *
47      * Returns a boolean value indicating whether the
48         operation succeeded.
49      *
50      * IMPORTANT: Beware that changing an allowance w
51         ith this method brings the risk
52      * that someone may use both the old and the new
53         allowance by unfortunate
54      * transaction ordering. One possible solution to
55         mitigate this race
56      * condition is to first reduce the spender's all
57         owance to 0 and set the
58      * desired value afterwards:

```

```

44 * https://github.com/ethereum/EIPs/issues/20#iss
uecomment-263524729
45 *
46 * Emits an {Approval} event.
47 */
48 function approve(address spender, uint256 amount)
external returns (bool);
49
50 /**
51 * @dev Moves `amount` tokens from `sender` to `r
ecipient` using the
52 * allowance mechanism. `amount` is then deducted
from the caller's
53 * allowance.
54 *
55 * Returns a boolean value indicating whether the
operation succeeded.
56 *
57 * Emits a {Transfer} event.
58 */
59 function transferFrom(address sender, address rec
ipient, uint256 amount) external returns (bool);
60
61 /**
62 * @dev Emitted when `value` tokens are moved fro
m one account (`from`) to
63 * another (`to`).
64 *
65 * Note that `value` may be zero.
66 */
67 event Transfer(address indexed from, address inde
xed to, uint256 value);
68
69 /**
70 * @dev Emitted when the allowance of a `spender`
for an `owner` is set by
71 * a call to {approve}. `value` is the new allowa
nce.
72 */
73 event Approval(address indexed owner, address ind
exed spender, uint256 value);
74 }
75
76 interface IStaking {
77     function stake( uint _amount, address _recipien
t ) external returns ( bool );
78     function claim( address _recipient ) external;
79 }
80
81 contract StakingHelper {
82
83     event LogStake(address indexed recipient, uint
amount);
84
85     IStaking public immutable staking;
86     IERC20 public immutable Time;
87
88     constructor ( address _staking, address _Time )
{
89         require( _staking != address(0) );
90         staking = IStaking(_staking);
91         require( _Time != address(0) );
92         Time = IERC20(_Time);
93     }
94
95     function stake( uint _amount, address recipient
) external {
96         Time.transferFrom( msg.sender, address(thi
s), _amount );

```

```

45 * https://github.com/ethereum/EIPs/issues/20#iss
uecomment-263524729
46 *
47 * Emits an {Approval} event.
48 */
49 function approve(address spender, uint256 amount)
external returns (bool);
50
51 /**
52 * @dev Moves `amount` tokens from `sender` to `r
ecipient` using the
53 * allowance mechanism. `amount` is then deducted
from the caller's
54 * allowance.
55 *
56 * Returns a boolean value indicating whether the
operation succeeded.
57 *
58 * Emits a {Transfer} event.
59 */
60 function transferFrom(address sender, address rec
ipient, uint256 amount) external returns (bool);
61
62 /**
63 * @dev Emitted when `value` tokens are moved fro
m one account (`from`) to
64 * another (`to`).
65 *
66 * Note that `value` may be zero.
67 */
68 event Transfer(address indexed from, address inde
xed to, uint256 value);
69
70 /**
71 * @dev Emitted when the allowance of a `spender`
for an `owner` is set by
72 * a call to {approve}. `value` is the new allowa
nce.
73 */
74 event Approval(address indexed owner, address ind
exed spender, uint256 value);
75 }
76
77 interface IStaking {
78     function stake( uint _amount, address _recipien
t ) external returns ( bool );
79     function claim( address _recipient ) external;
80 }
81
82 contract StakingHelper {
83
84     event LogStake(address indexed recipient, uint
amount);
85
86     IStaking public immutable staking;
87     IERC20 public immutable Time;
88
89     constructor ( address _staking, address _Time )
{
90         require( _staking != address(0) );
91         staking = IStaking(_staking);
92         require( _Time != address(0) );
93         Time = IERC20(_Time);
94     }
95
96     function stake( uint _amount, address recipient
) external {
97         Time.transferFrom( msg.sender, address(thi
s), _amount );

```

```
97     Time.approve( address(staking), _amount );
98     staking.stake( _amount, recipient );
99     staking.claim( recipient );
100     emit LogStake(recipient, _amount);
101 }
102 }
```

```
98     Time.approve( address(staking), _amount );
99     staking.stake( _amount, recipient );
100    staking.claim( recipient );
101    emit LogStake(recipient, _amount);
102 }
103 }
```

