Solidity Generated code.sol

```
pragma solidity ^0.8;
// SPDX-License-Identifier: MIT

contract B_weth
{

    uint private constant threshold = 2;

    mapping (address => uint) private accountOf;
    mapping (address => mapping (address => uint)) private allowanceOf;
    mapping (address => bool) private depositedOver_100;
    address private manager;
    uint private index;
    address[] private depositors;
    bool private donated;

    constructor (){
      index = 0;
      manager = msg.sender;
      donated = false;
    }

    function deposit ( ) payable public {
      uint senderAccount = accountOf[msg.sender];
      uint senderBalance = msg.sender.balance;
      uint thisBalance = address(this).balance;
      if ( thisBalance+msg.value<=type(uint).max&&senderBalance-msg.value>=0&&
senderAccount+msg.value<=type(uint).max ){
        accountOf[msg.sender] = senderAccount+msg.value;
        bool distinct = depositedOver_100[msg.sender];
        if ( senderAccount+msg.value>=threshold&&distinct==false&&index<threshold ){
          depositors.push(msg.sender);
          depositedOver_100[msg.sender] = true;
          index = index+1;
        }
      }
      else {
        revert();
      }
    }

    function withdraw (uint amount ) public {
      uint senderAccount = accountOf[msg.sender];
      uint senderBalance = msg.sender.balance;
      uint thisBalance = address(this).balance;
      if ( senderAccount>=amount&&senderBalance+amount<=type(uint).max&&thisBalance>=
amount ){
        payable(msg.sender).transfer(amount);
        accountOf[msg.sender] = senderAccount-amount;
      }
      else {
        revert();
      }
    }

    function transferTo (uint amount, address dst ) public {
```

```
55        uint senderBalance = accountOf[msg.sender];
56        uint receiverBalance = accountOf[dst];
57        if ( senderBalance>amount&&receiverBalance+amount<=type(uint).max&&msg.sender!=
dst ){
58          accountOf[msg.sender] = senderBalance-amount;
59          accountOf[dst] = receiverBalance+amount;
60        }
61        else {
62          revert();
63        }
64      }
65
66      function approve (uint amount, address dst ) public {
67        if ( msg.sender!=dst ){
68          allowanceOf[msg.sender][dst] = amount;
69        }
70        else {
71          revert();
72        }
73      }
74
75      function transferFrom (uint amount, address sender, address recipient ) public {
76        uint senderBalance = accountOf[sender];
77        uint recipientBalance = accountOf[recipient];
78        uint allowance = allowanceOf[sender][msg.sender];
79        if ( sender!=recipient&&allowance>=amount&&senderBalance>=amount&&
recipientBalance+amount<=type(uint).max ){
80          accountOf[sender] = senderBalance-amount;
81          accountOf[recipient] = recipientBalance+amount;
82          allowanceOf[sender][msg.sender] = allowance-amount;
83        }
84        else {
85          revert();
86        }
87      }
88
89      function rewardTopDepositors ( ) payable public {
90        uint thisBalance = address(this).balance;
91        uint managerBalance = manager.balance;
92        if ( msg.value==threshold&&msg.sender==manager&&index==threshold&&donated==false&
&thisBalance+msg.value<=type(uint).max&&managerBalance-msg.value>=0 ){
93          uint jj;
94          bool safe;
95          jj = 0;
96          safe = true;
97          while(jj<index&&safe==true){
98            uint depositorBalance = accountOf[depositors[jj]];
99            safe = depositorBalance+1<=type(uint).max;
100           jj = jj+1;
101         }
102         if ( ( safe==true ) ){
103           donated = true;
104           uint ii;
105           ii = 0;
106           while(ii<index){
107             uint depositorBalance = accountOf[depositors[ii]];
108             accountOf[depositors[ii]] = depositorBalance+1;
109             ii = ii+1;
110           }
```

```
111           }
112         }
113         else {
114           revert();
115         }
116       }
117
118 }
119
```