Blockchain Technology

D2

19BCE248

Practical 5

AIM: Implementing Smart Contract for your respective Term Paper topic

Topic: Smart Parking System

Code:

```
pragma solidity ^0.5.0;
contract SmartParking{
    struct Parking{
        bool isParked;
        uint price;
        bool isSpecial;
    }
    struct Person{
        bool isGuest;
        bool isAuthenticated;
    }

    Parking[] public parking;
    Person[] public person;

    function addPerson(bool isGuest,bool isAuthenticated) public {
        person.push(Person(isGuest,isAuthenticated));
    }

    function addParking(bool isParked,uint price,bool isSpecial) public {
        parking.push(Parking(isParked,price,isSpecial));
    }

    function getVacantPlace(bool isGuest) public view returns(uint){
            for(uint i=0;i<parking.length;i++){
                    if(!parking[i].isParked ){
                        if(isGuest && !parking[i].isSpecial){
                            return i;
                        }else if(!isGuest && parking[i].isSpecial){
                            return i;
                        }
                    }
            }
            return 1000;
    }
    function allocateParking(bool isGuest, bool isAuthenticated) public 
returns(bool){
        require(isAuthenticated);
        uint slotIndex=getVacantPlace(isGuest);
        if(slotIndex==1000){
            return false;
        }
        parking[slotIndex].isParked=true;
    }
}
```

Here we have implemented a very basic level of smart contract which covers most of the concept of solidity and to get familiar with the syntax as well as the remix environment.

There are mainly two stakeholders:

1) Person (Consumer)
    - isGuest : Tells information about whether the particular person is daily visitor or guest.
    - isAuthenticated: A very essential part to see whether he/she has a valid existential or are bind with some inappropriate activities which could further lead some problem to owner.
2) Parking (Supplier)
    - isParked: Occupied/Vacant
    - price: amount of parking slot
    - isSpecial: Already reserved for daily customer or for the owners or pre-booked.

Similary we have three functions namely:

1) addPerson : Called whenever a new person arrives at entry gate
2) addParking : Pre-defined at the time of construction
3) getVacantPlace : According to current snapshot return index of vacant places
4) allocateParking : getVacantPlace check for authenticity and allocate parking if all the requirements fulfilled.

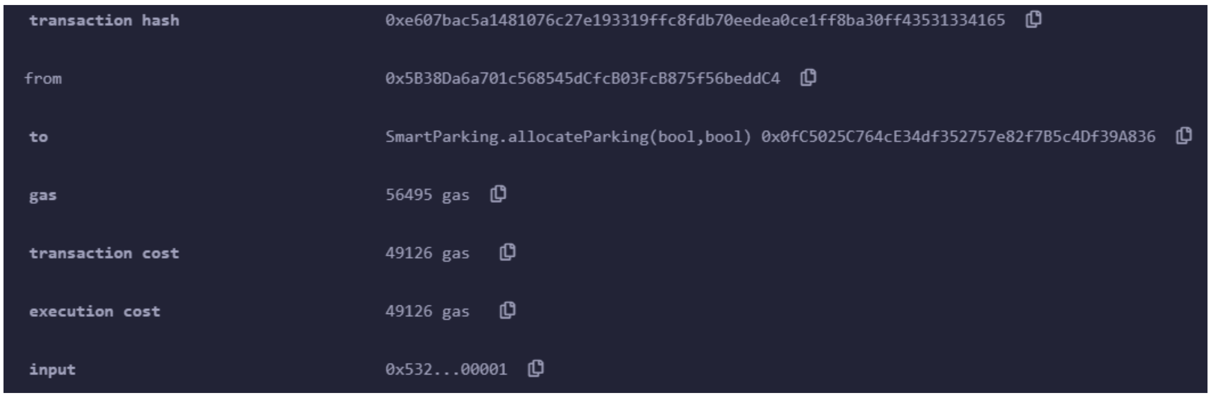
Output:

Parking:

1) false,35,false
2) false,35,false
3) false,35,false
4) false,50,true
5) false,50,true

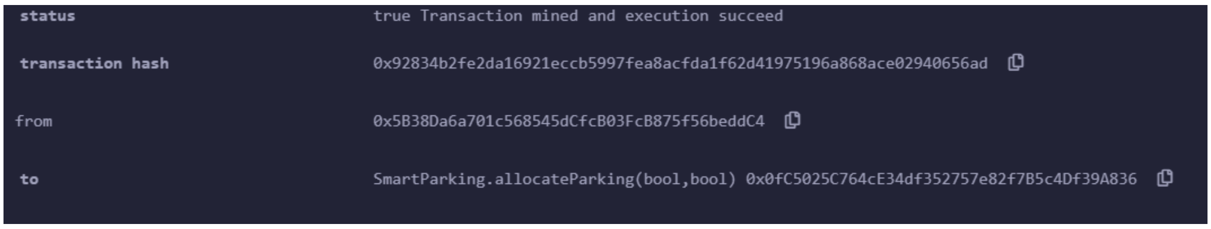
allocateParking:

1) true,true



```
transaction hash      0xe607bac5a1481076c27e193319ffc8fdb70eedea0ce1ff8ba30ff43531334165
from                  0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                    SmartParking.allocateParking(bool,bool) 0x0fC5025C764cE34df352757e82f7B5c4Df39A836
gas                   56495 gas
transaction cost      49126 gas
execution cost        49126 gas
input                 0x532...00001
```

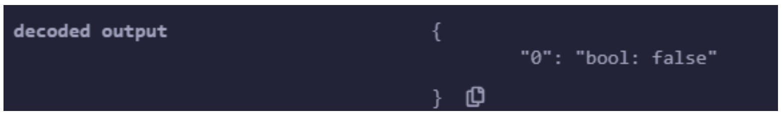
2) true,true



```
status                true Transaction mined and execution succeed
transaction hash      0x92834b2fe2da16921eccb5997fea8acfda1f62d41975196a868ace02940656ad
from                  0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                    SmartParking.allocateParking(bool,bool) 0x0fC5025C764cE34df352757e82f7B5c4Df39A836
```

3) true,true
Will again allocate

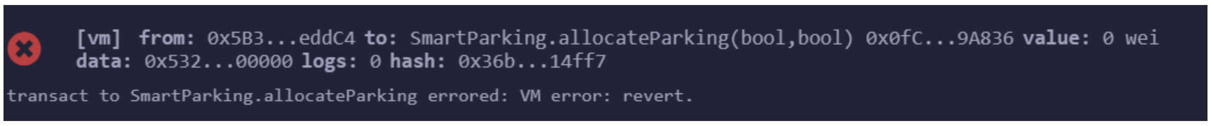
4) true,true



```
decoded output        {
                            "0": "bool: false"
                      }
```

Deny

5) false,false



```
[vm] from: 0x5B3...eddC4 to: SmartParking.allocateParking(bool,bool) 0x0fC...9A836 value: 0 wei
data: 0x532...00000 logs: 0 hash: 0x36b...14ff7
transact to SmartParking.allocateParking errored: VM error: revert.
```

Error due to not authenticate person