

انتخابات شورای صنفی دانشجویی دانشگاه شهید بهشتی

نگین مشایخی ۹۸۲۴۳۰۵۴ - پارسا نوری ۹۸۲۴۳۰۶۷

این کد شامل تعدادی تابع برای پیاده سازی بخش های مورد نظر است. پروتوتایپ تابع ها در زیر آمده است:

```
function group_permission(address[] memory vs, uint _size) public{ ...
}

//this function can be called from chairman to give permission to an addresses
function give_permission(address voter) public{ ...
}

//this function is for voting
function vote(string memory _vote) public { ...
}

function transfer_chance(address next_voter) public{ ...
}

function result() public view returns (string memory){ ...
}

//this function return time - it is for debugging part
function Time_call() public view returns (uint){ ...
}

//chairman can extend the time of the voting
function time_extention(uint _new_time) public{ ...
}
```

- تابع **group permission** لیستی از آدرس ها را می گیرد و طبق شرایط زیر اجازه شرکت در رای گیری را به افراد داخل لیست میدهد.
 - این تابع باید در زمان رای گیری صدا زده شود.
 - این تابع باید توسط مسئول انتخابات صدا زده شود.
 - تعداد افراد جدید + تعداد افراد دارای حق رای باید از حد مجاز تعیین شده تجاوز نکند.
 - هیچ یک از افراد داخل لیست نباید از قبل حق رای داشته باشند.

```

function group_permission(address[] memory vs, uint _size) public{
    //should be in valid time
    require(
        block.timestamp < end_time && block.timestamp>start_time,
        "Voting is over or not started yet"
    );
    //sender must be chairman
    require(
        msg.sender == chairman,
        "only chairman can give permission to voter."
    );
    //number of permissions should be check,there is a limit for it.
    require(
        limit_votes >= permission_counts+_size,
        "Vote limit is completed."
    );
    //each address in list should be a new address who wasnt permitted yet
    for(uint i=0;i<_size;i++){
        require(
            !voters[vs[i]].can_vote,
            "Voter has already have permission."
        );
    }
    for(uint i=0;i<_size;i++){
        voters[vs[i]].can_vote = true;
        voters[vs[i]].chance = 1;
    }
    permission_counts = permission_counts + _size;
}

```

- تابع **give permission** مانند تابع بالاست با این تفاوت که برای یک نفر حق رای را صادر میکند.

```
//this function can be called from chairman to give permission to an addresses
function give_permission(address voter) public{
    require(
        block.timestamp < end_time && block.timestamp>start_time,
        "Voting is over or not started yet"
    );
    require(
        msg.sender == chairman,
        "only chairman can give permission to voter."
    );
    //the voter must have permission to vote
    require(
        !voters[voter].can_vote,
        "Voter has already have permission."
    );
    require(
        limit_votes>permission_counts,
        "Vote limit is completed."
    );
    permission_counts++;
    voters[voter].can_vote = true;
    voters[voter].chance = 1;
}
```

- تابع **vote** رای یک نفر را گرفته و ثبت میکند. شرایط مجاز بودن این رای به صورت زیر است:
 - این تابع باید در زمان رای گیری صدا زده شود.
 - رای دهنده حق رای از طرف مسئول برگزاری انتخابات را داشته باشد.
 - رای دهنده شانس رای دادن داشته باشد(قبلا رای نداده باشد یا حق رای از کسی به اون منتقل شده باشد).
 - فردی که به آن رای میدهد از بین کاندید های مجاز انتخابات باشد.
- در این صورت شانس فرد برای رای دادن از اون گرفته شده و رای او ثبت میشود.

```

//this function is for voting
function vote(string memory _vote) public {
    require(
        block.timestamp < end_time && block.timestamp>start_time,
        "Voting is over or not started yet"
    );
    Voter storage sender = voters[msg.sender];
    require(
        sender.can_vote,
        "The voter has no permission to vote!"
    );
    //voter must have chance to vote
    require(
        sender.chance>0,
        "This voter has already voted!"
    );
    //the person must be a candidate of the list
    bool in_list = false;
    for (uint i = 0; i<candidates.length;i++){
        if( keccak256(bytes(candidates[i])) == keccak256(bytes(_vote)) ){
            candidate_votes[i]++; //votes of the candidate increase
            sender.chance--; //loss 1 chance to vote
            vote_counts++; //number of all votes increase
            in_list = true; //that was a valid candidate
        }
    }
    require(
        in_list,
        "The person you chose is not a candidate"
    );
}
}

```

- تابع **transfer chance** از طرف رای دهنده ای صدا زده شده و ادرس رای دهنده مورد نظر را ورودی میگیرد. به این صورت که یک شانس رای از نفر اول گرفته شده و به نفر دوم منتقل میشود. شرایط مجاز بودن این انتقال به صورت زیر است:
 - این تابع باید در زمان رای گیری صدا زده شود.
 - رای دهنده اول حق رای از طرف مسئول برگزاری انتخابات را داشته باشد.
 - رای دهنده اول شانس رای دادن داشته باشد.
 - رای دهنده دوم حق رای از طرف مسئول برگزاری انتخابات را داشته باشد.

```

function transfer_chance(address next_voter) public{
    Voter storage sender = voters[msg.sender];
    require(
        block.timestamp < end_time && block.timestamp > start_time,
        "Voting is over or not started yet"
    );
    require(
        sender.can_vote,
        "The voter has no permission to vote!"
    );
    require(
        sender.chance > 0,
        "This voter has already voted!"
    );
    Voter storage reciever = voters[next_voter];
    require(
        reciever.can_vote,
        "The reciever has no permission to vote!"
    );
    reciever.chance++; //reciever has second chance to vote
    sender.chance--; //sender have no long any chance to vote
}

```

- تابع **result** در انتهای انتخابات نام کاندید منتخب را اعلام میکند. در صورت برابری رای کاندیداهای منتخب هیچکس برنده انتخابات نیست و در به حد نصاب نرسیدن آرا انتخابات ملغی اعلام میشود. این تابع فقط زمانی که انتخابات به پایان زمان خود رسیده قابل اجرا است.

```

function result() public view returns (string memory){
    //voting must be ended to see result
    require(
        block.timestamp >= end_time,
        "Voting time is not over yet"
    );
    //if less than half voted the voting will be canceled
    if(vote_counts < limit_votes/2){
        return "CANCELED";
    }
    //find the winner candidate
    uint max_count;
    string memory max_name;
    bool same_count;
    for (uint i = 0; i<candidates.length;i++){
        if( candidate_votes[i] == max_count){
            same_count = true;
        }else if(candidate_votes[i] > max_count){
            same_count = false;
            max_count = candidate_votes[i];
            max_name = candidates[i];
        }
    }
    //if some of candidates have same number of votes there is no winner
    if(same_count){
        return "NO_WINNER";
    }else{
        return max_name;
    }
}

```

- تابع **time extension** زمان پایان انتخابات را تمدید میکند. باید توسط مسئول انتخابات صدا زده شود و زمان تمدید شده از زمان فعلی دیرتر باشد.






```

//chairman can extend the time of the voting
function time_extention(uint _new_time) public{
    require(
        msg.sender == chairman,
        "Only chairman can stop voting!"
    );
    //the new end-time should be after the old one
    require(
        _new_time>end_time,
        "previous end time > the new end time"
    );
    end_time = _new_time;
}


```



برای دیپلوی در شبکه تست بلاک چین پس از کامپایل کد تنظیمات Remix IDE را در حالت تصویر زیر قرار میدهیم.

پارامترهای TITLE , LIMIT , START , END , CANDIDATE LIST پارامترهای Constructor هستند. اکانتی که با آن constructor را صدا میزنیم به عنوان اکانت مسئول انتخابات ست میشود. بعد از deploy میتوانیم گزارش transaction ها و آدرس ها را در Ganache مشاهده کنیم.







DEPLOY & RUN TRANSACTIONS

ENVIRONMENT 

Ganache Provider  

Custom (5777) network

ACCOUNT 


0xc27...d0E85 (99.3094813 ether)   

GAS LIMIT


3000000


VALUE

0

Wei 

CONTRACT (Compiled by Remix)

Voting - contracts/4_Voting.sol 

DEPLOY 



_TITLE: shora senfi

_LIMIT: 4

_START: 1668757399

_END: 1668857399

_CANDIDATE_LIST: ["amir","sara","ali"]

 Calldata  Parameters

transact

☐ Publish to IPFS

OR

At Address

Load contract from Address