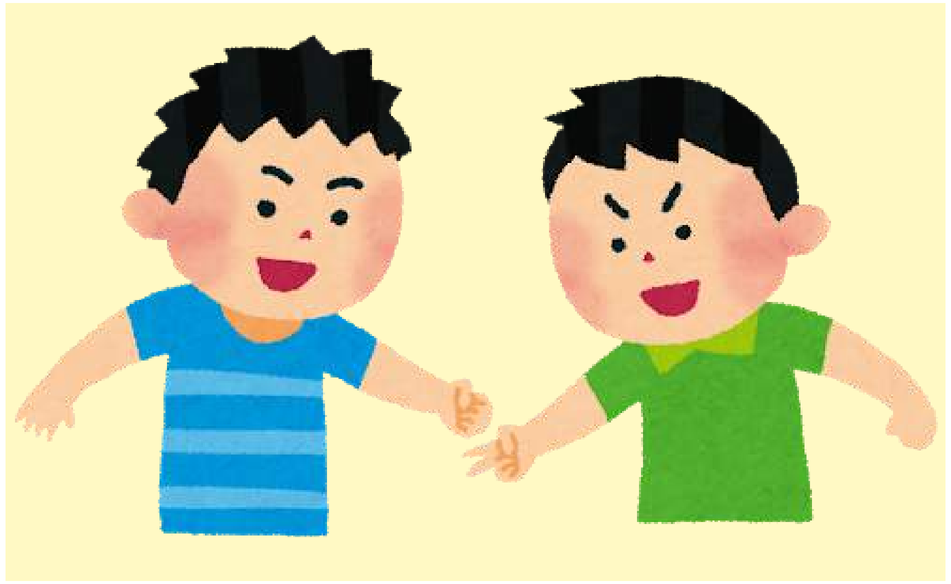
Let's Practice Scilla

Gauidy inc.

じゃんけんコントラクトを作ってみよう



- **穴埋め形式でScillaを覚えていきましょう**

  ソースは以下のリポジトリからダウンロードしてください

  https://github.com/tky5622/scilla-practical-workshop

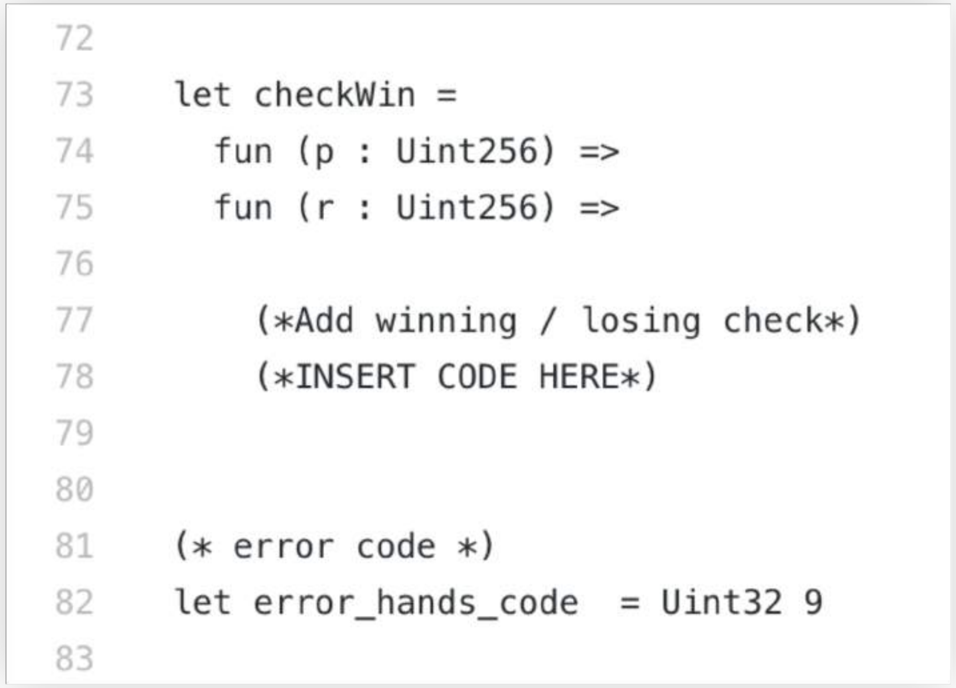
- **Savant-IDEを使っていきましょう**

  https://savant-ide.zilliqa.com/

  コードチェックからデプロイテストまで実行可能

  Scillaで書かれた代表的なコントラクト付き

進め方



```

let checkWin =
  fun (p : Uint256) =>
  fun (r : Uint256) =>

    (*Add winning / losing check*)
    (*INSERT CODE HERE*)


(* error code *)
let error_hands_code  = Uint32 9

```

- **各お題をデプロイまで実行してみよう!**
  1. じゃんけん判定を書いてみよう
  2. ランダムな対戦相手を作成してみよう
  3. 賭け金の仕組みをいれてみよう
  4. コントラクトを分割してみよう

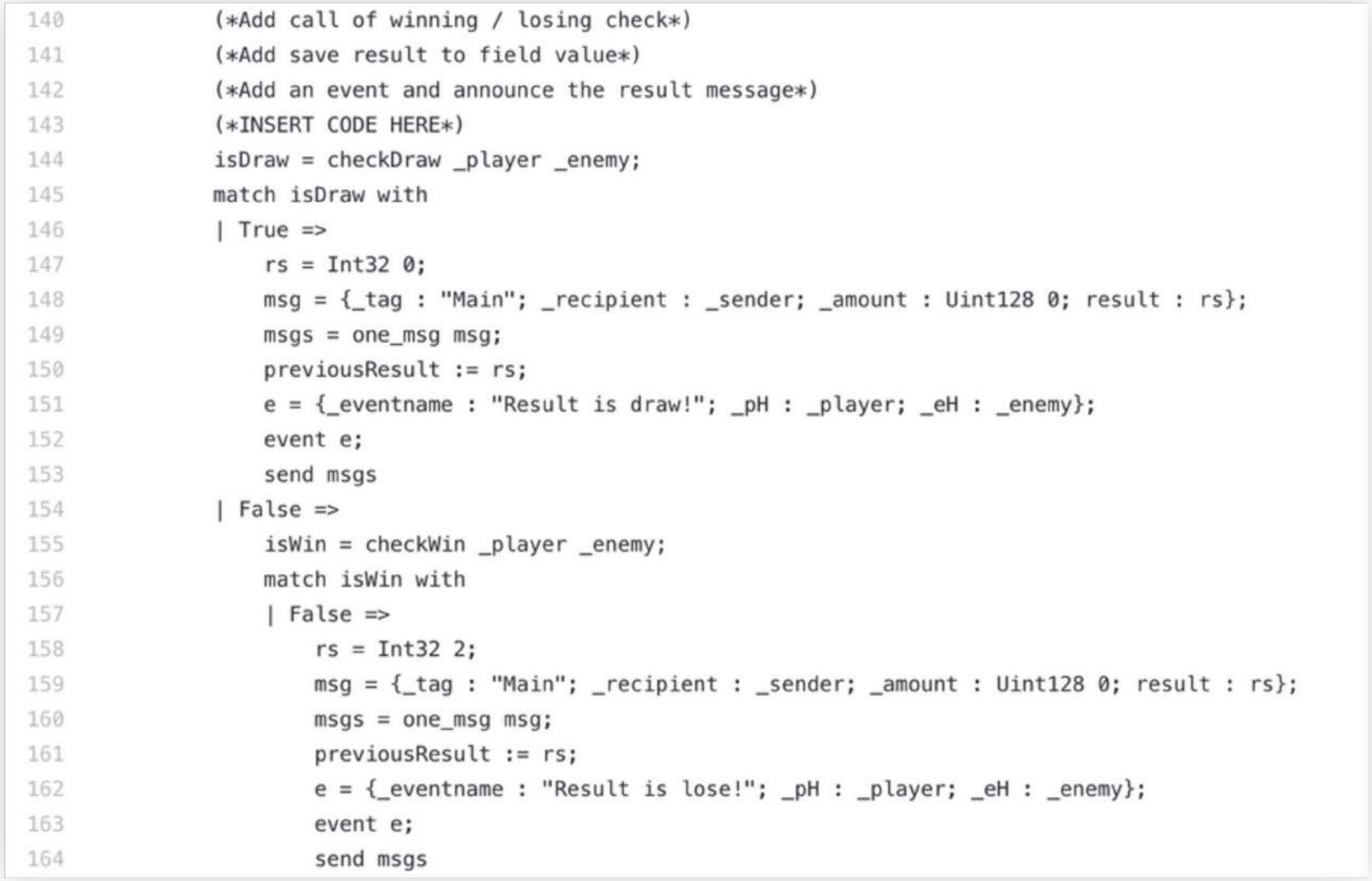
参考資料

- Scilla-doc
- Built-in Dictionary
- Gitter-Zilliqa Smart Contract
- Scilla-vanilla

Let's Practice

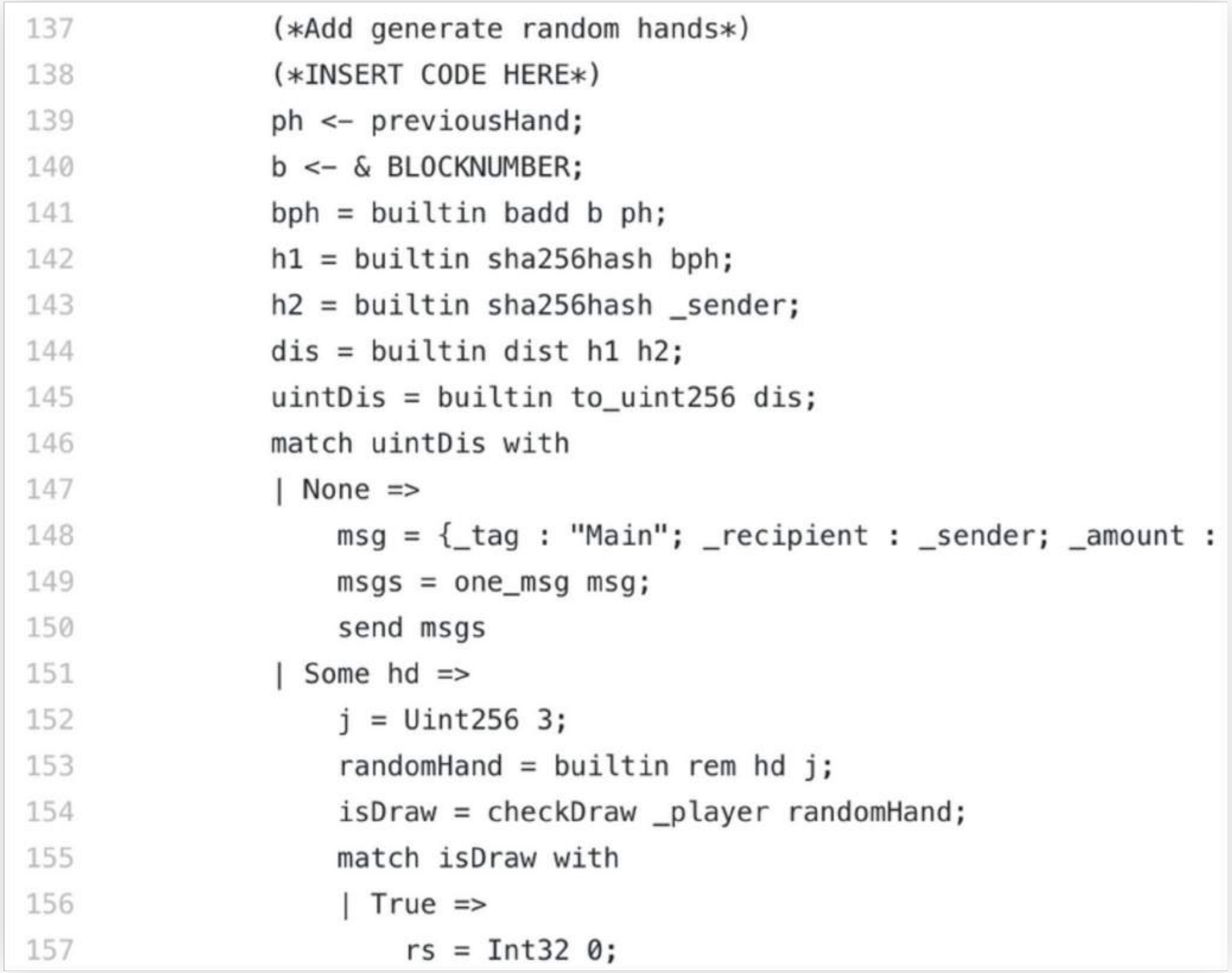
ご不明な点がございましたら随時質問してください

Practice1 Answer



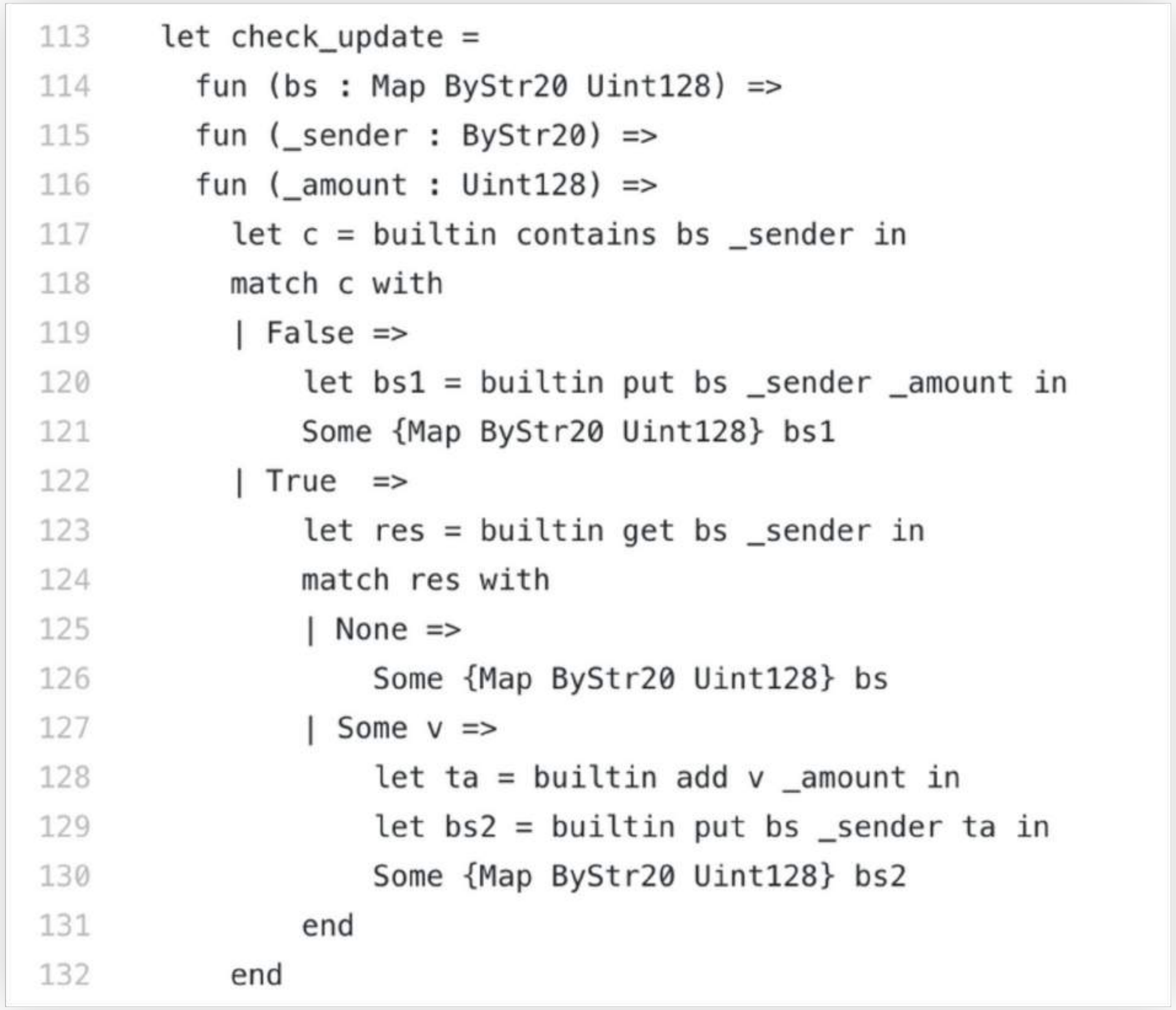
```
(*Add call of winning / losing check*)
(*Add save result to field value*)
(*Add an event and announce the result message*)
(*INSERT CODE HERE*)
isDraw = checkDraw _player _enemy;
match isDraw with
| True =>
    rs = Int32 0;
    msg = {_tag : "Main"; _recipient : _sender; _amount : Uint128 0; result : rs};
    msgs = one_msg msg;
    previousResult := rs;
    e = {_eventname : "Result is draw!"; _pH : _player; _eH : _enemy};
    event e;
    send msgs
| False =>
    isWin = checkWin _player _enemy;
    match isWin with
    | False =>
        rs = Int32 2;
        msg = {_tag : "Main"; _recipient : _sender; _amount : Uint128 0; result : rs};
        msgs = one_msg msg;
        previousResult := rs;
        e = {_eventname : "Result is lose!"; _pH : _player; _eH : _enemy};
        event e;
        send msgs
```

Practice2 Answer



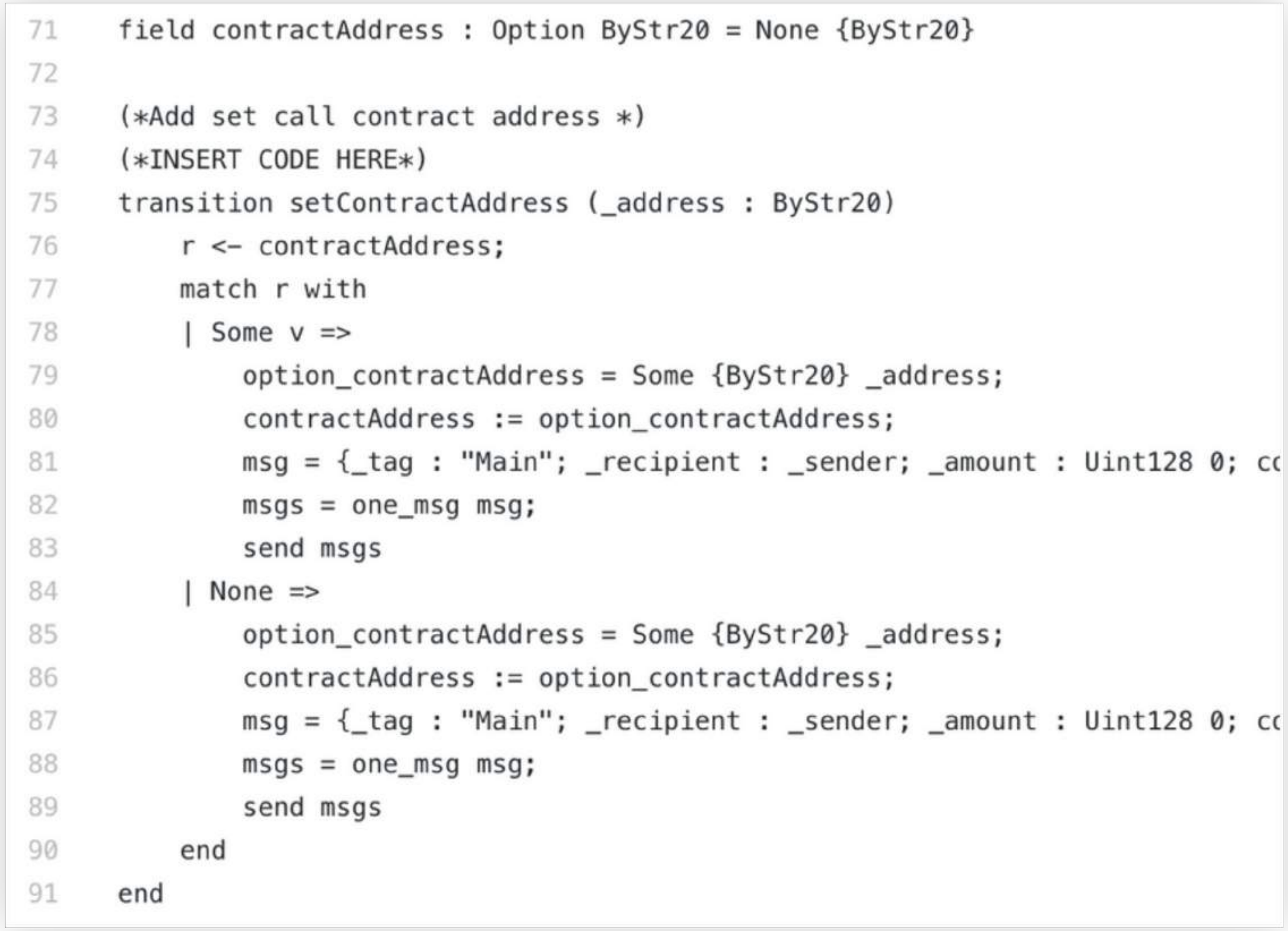
```
(*Add generate random hands*)
(*INSERT CODE HERE*)
ph <- previousHand;
b <- & BLOCKNUMBER;
bph = builtin badd b ph;
h1 = builtin sha256hash bph;
h2 = builtin sha256hash _sender;
dis = builtin dist h1 h2;
uintDis = builtin to_uint256 dis;
match uintDis with
| None =>
    msg = {_tag : "Main"; _recipient : _sender; _amount :
    msgs = one_msg msg;
    send msgs
| Some hd =>
    j = Uint256 3;
    randomHand = builtin rem hd j;
    isDraw = checkDraw _player randomHand;
    match isDraw with
    | True =>
        rs = Int32 0;
```

Practice3 Answer



```
let check_update =
  fun (bs : Map ByStr20 Uint128) =>
  fun (_sender : ByStr20) =>
  fun (_amount : Uint128) =>
    let c = builtin contains bs _sender in
    match c with
    | False =>
        let bs1 = builtin put bs _sender _amount in
        Some {Map ByStr20 Uint128} bs1
    | True  =>
        let res = builtin get bs _sender in
        match res with
        | None =>
            Some {Map ByStr20 Uint128} bs
        | Some v =>
            let ta = builtin add v _amount in
            let bs2 = builtin put bs _sender ta in
            Some {Map ByStr20 Uint128} bs2
        end
    end
```

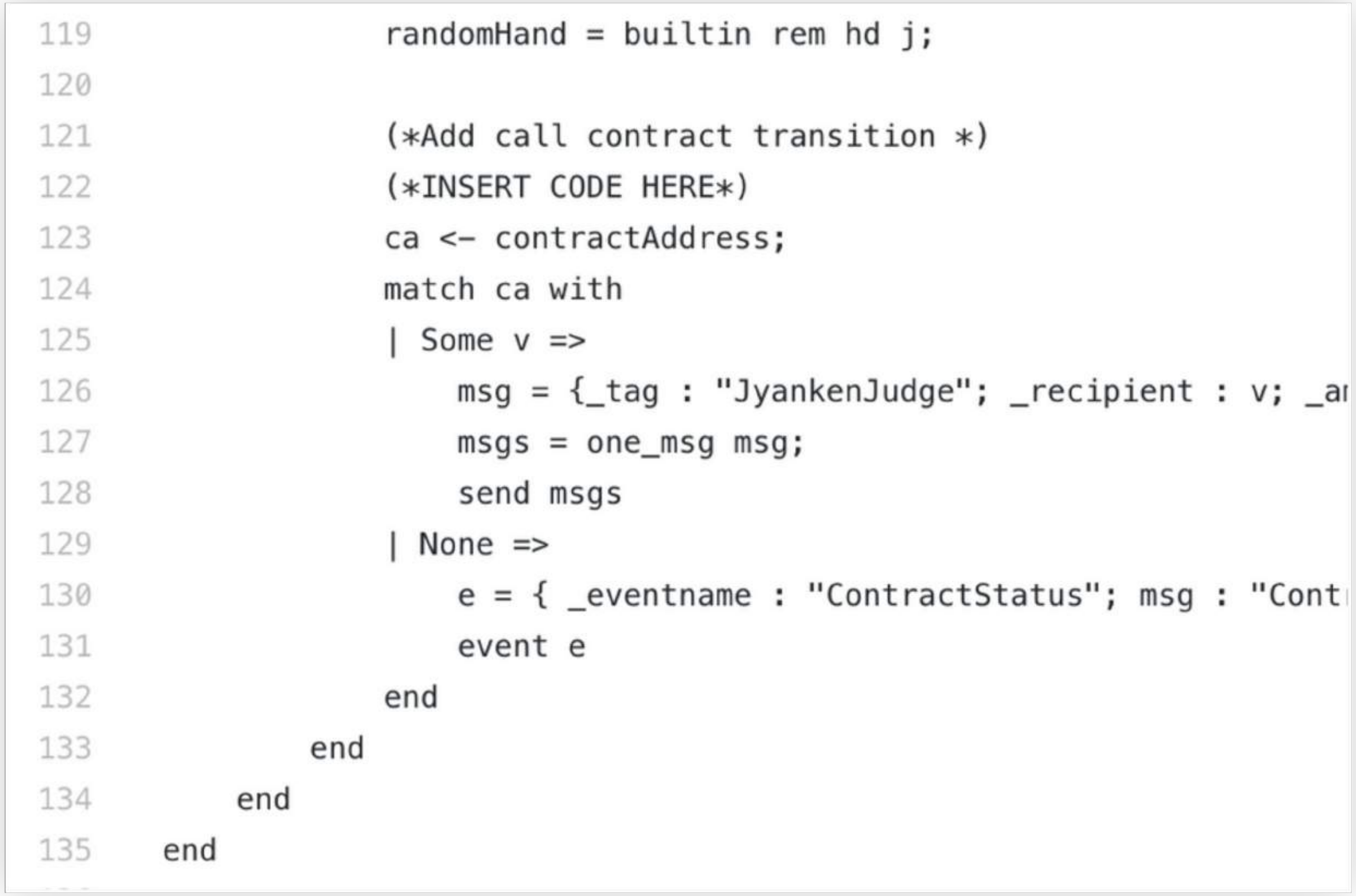
Practice4-1 Answer



```
field contractAddress : Option ByStr20 = None {ByStr20}

(*Add set call contract address *)
(*INSERT CODE HERE*)
transition setContractAddress (_address : ByStr20)
    r <- contractAddress;
    match r with
    | Some v =>
        option_contractAddress = Some {ByStr20} _address;
        contractAddress := option_contractAddress;
        msg = {_tag : "Main"; _recipient : _sender; _amount : Uint128 0; cc
        msgs = one_msg msg;
        send msgs
    | None =>
        option_contractAddress = Some {ByStr20} _address;
        contractAddress := option_contractAddress;
        msg = {_tag : "Main"; _recipient : _sender; _amount : Uint128 0; cc
        msgs = one_msg msg;
        send msgs
    end
end
```

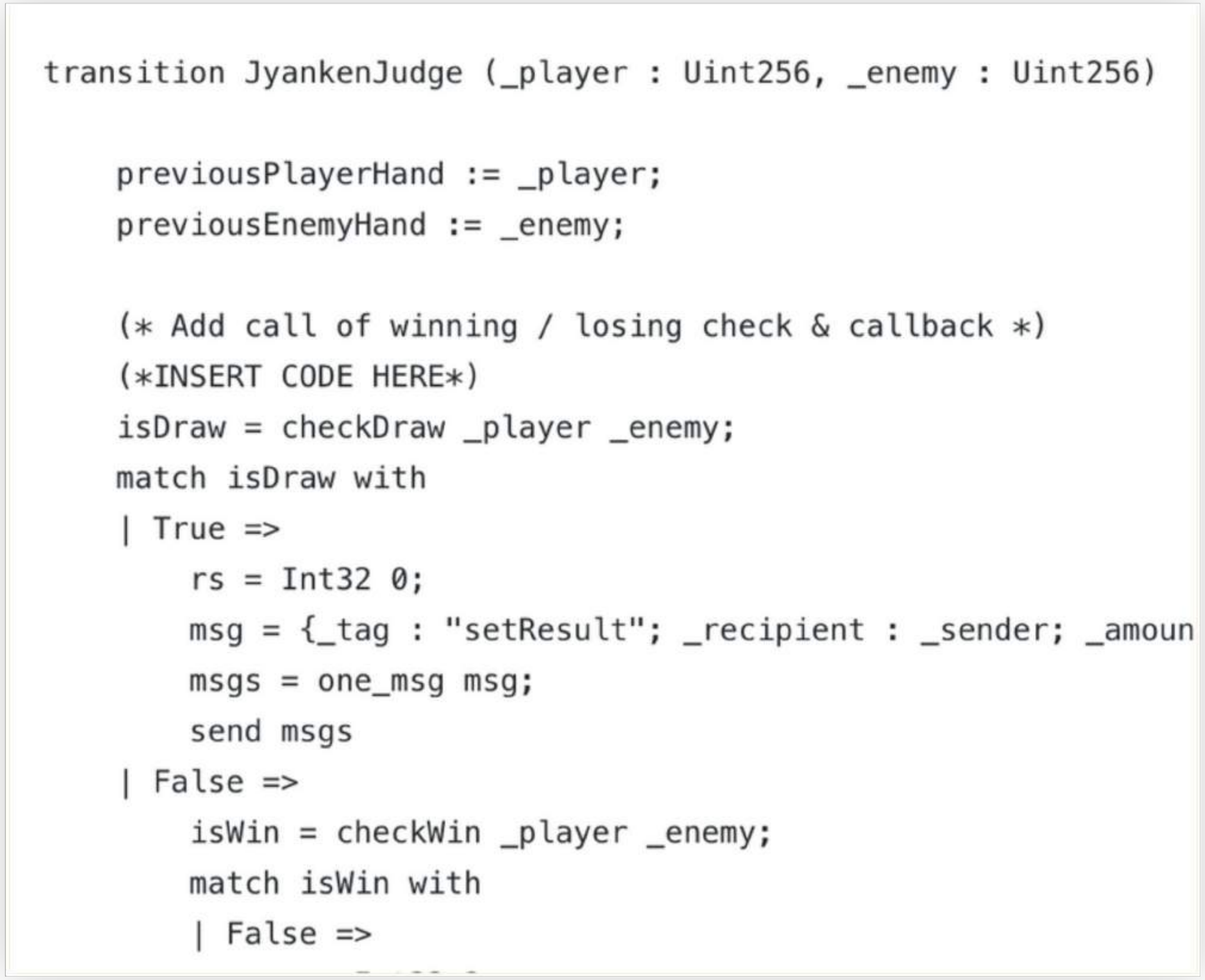
Practice4-2 Answer



```
                randomHand = builtin rem hd j;

                (*Add call contract transition *)
                (*INSERT CODE HERE*)
                ca <- contractAddress;
                match ca with
                | Some v =>
                    msg = {_tag : "JyankenJudge"; _recipient : v; _ar
                    msgs = one_msg msg;
                    send msgs
                | None =>
                    e = { _eventname : "ContractStatus"; msg : "Cont
                    event e
                end
            end
        end
    end
```

Practice4-3 Answer



```
transition JyankenJudge (_player : Uint256, _enemy : Uint256)

    previousPlayerHand := _player;
    previousEnemyHand := _enemy;

    (* Add call of winning / losing check & callback *)
    (*INSERT CODE HERE*)
    isDraw = checkDraw _player _enemy;
    match isDraw with
    | True =>
        rs = Int32 0;
        msg = {_tag : "setResult"; _recipient : _sender; _amoun
        msgs = one_msg msg;
        send msgs
    | False =>
        isWin = checkWin _player _enemy;
        match isWin with
        | False =>
```

Thanks for listening.

Gaudiy will further pursue the scilla.