```
def evalState(event, beforeState):
    #offencePlayerLst는 event에서 나온 각 주자가 어디로 갔는지에 대한 정보.
    #home으로 들어온 선수는 4번 칸에 들어가므로 3번칸 까지 잘라준다.
    #이런 방식이 아니더라도, event class의 정보를 이용해서 새로운 offense team을 
계산해주는 과정이 반드시 필요함.
    offensePlayerLst = [0,0,0,0,0]
    offensePlayerLst[event.getHitter()] = 
beforeState.getOffenseTeam().getOffenseTeamLst()[0]
    offensePlayerLst[event.getB1runner()] = 
beforeState.getOffenseTeam().getOffenseTeamLst()[1]
    offensePlayerLst[event.getB2runner()] = 
beforeState.getOffenseTeam().getOffenseTeamLst()[2]
    offensePlayerLst[event.getB3runner()] = 
beforeState.getOffenseTeam().getOffenseTeamLst()[3]
    offensePlayerLst = offensePlayerLst[:4]


    #다음 타자에 대한 정보가 필요할 수 있는데.....
    switch (event)
        #single
        case single :
            afterState = State()
            #after state를 정의하려면 다음 타자에 대한 정보가 필요하다.
            #도루와 같은 경우에는 runner들의 정보는 바뀌지만 Hitter는 바뀌지 않으
므로 반드시 다음 타자는 아니다.
            offencePlayerLst[0] = "nextStateHitter"
            afterState.setState(offencePlayerLst, 
beforeState.getDefenseTeam().getDefenseTeamLst(), 0, 
beforeState.getOut(), 0, beforeState.getTurn(), 
beforeState.getIsHome())
            return afterState

        case double :
            afterState = State()
            offencePlayerLst[0] = "nextStateHitter"
            afterState.setState(offencePlayerLst, 
beforeState.getDefenseTeam().getDefenseTeamLst(), 0, 
beforeState.getOut(), 0, beforeState.getTurn(), 
beforeState.getIsHome())
            return afterState

        case triple :
            afterState = State()
            offencePlayerLst[0] = "nextStateHitter"
            afterState.setState(offencePlayerLst, 
beforeState.getDefenseTeam().getDefenseTeamLst(), 0, 
beforeState.getOut(), 0, beforeState.getTurn(), 
beforeState.getIsHome())
            return afterState

        case homerun :
            afterState = State()
            offencePlayerLst[0] = "nextStateHitter"
            afterState.setState(offensePlayerLst,
beforeState.getDefenseTeam().getDefenseTeamLst(), 0,
beforeState.getOut(), 0, beforeState.getTurn(),
beforeState.getIsHome())
            return afterState

        case error :
            afterState = State()
            offencePlayerLst[0] = "nextStateHitter"
            afterState.setState(offensePlayerLst,
beforeState.getDefenseTeam().getDefenseTeamLst(), 0,
beforeState.getOut(), 0, beforeState.getTurn(),
beforeState.getIsHome())
            return afterState

        case foul :
            afterState = State()
            offencePlayerLst[0] =
beforeState.getOffenseTeam().getOffenseTeamLst()[0]
            if beforeState.getStrike() != 2:

afteState.setState(beforeState.getOffenseTeam().getOffenseTeamLst(),
beforeState.getDefenseTeam().getDefenseTeamLst(),
beforeState.getBall(), beforeState.getOut(), beforeState.getStrike()
+ 1, beforeState.getTurn(), beforeState.getIsHome())
            return afterState

        case singleOut :
            #out을 통해서 만약 공수교대가 된다면 이때는 State를 어떻게 할까? 어차
피 state가 공수교대로 넘어가니까 상관이 없으려나?
            #그리고 공수교대 될때 다음 타자가 누구였는지를 저장해야할것같다.
            afterState = State()
            offencePlayerLst[0] = "nextStateHitter"
            afterState.setState(offensePlayerLst,
beforeState.getDefenseTeam().getDefenseTeamLst(), 0,
beforeState.getOut() + 1, 0, beforeState.getTurn(),
beforeState.getIsHome())
            return afterState

        case doublePlay :
            afterState = State()
            offencePlayerLst[0] = "nextStateHitter"
            afterState.setState(offensePlayerLst,
beforeState.getDefenseTeam().getDefenseTeamLst(), 0,
beforeState.getOut() + 2, 0, beforeState.getTurn(),
beforeState.getIsHome())
            return afterState

        case triplePlay :
            afterState = State()
            offencePlayerLst[0] = "nextStateHitter"
            afterState.setState(offensePlayerLst,
beforeState.getDefenseTeam().getDefenseTeamLst(), 0,
beforeState.getOut() + 3, 0, beforeState.getTurn(),
beforeState.getIsHome())
            return afterState

        case secrificeFly :
            afterState = State()
            offencePlayerLst[0] = "nextStateHitter"
            afterState.setState(offensePlayerLst,
beforeState.getDefenseTeam().getDefenseTeamLst(), 0,
beforeState.getOut() + 1, 0, beforeState.getTurn(),
beforeState.getIsHome())
            return afterState

        case secrificeBunt :
            afterState = State()
            offencePlayerLst[0] = "nextStateHitter"
            afterState.setState(offensePlayerLst,
beforeState.getDefenseTeam().getDefenseTeamLst(), 0,
beforeState.getOut() + 1, 0, beforeState.getTurn(),
beforeState.getIsHome())
            return afterState

        case swing :
            afterState = State()
            offencePlayerLst[0] =
beforeState.getOffenseTeam().getOffenseTeamLst()[0]

afteState.setState(beforeState.getOffenseTeam().getOffenseTeamLst(),
beforeState.getDefenseTeam().getDefenseTeamLst(),
beforeState.getBall(), beforeState.getOut(), beforeState.getStrike()
+ 1, beforeState.getTurn(), beforeState.getIsHome())
            return afterState

        case strikeOut :
            afterState = State()
            offencePlayerLst[0] = "nextStateHitter"
            afterState.setState(offensePlayerLst,
beforeState.getDefenseTeam().getDefenseTeamLst(), 0,
beforeState.getOut() + 1, 0, beforeState.getTurn(),
beforeState.getIsHome())
            return afterState

        case wildPitch :
            afterState = State()
            offencePlayerLst[0] = "nextStateHitter"
            afterState.setState(offensePlayerLst,
beforeState.getDefenseTeam().getDefenseTeamLst(), 0,
beforeState.getOut(), 0, beforeState.getTurn(),
beforeState.getIsHome())
            return afterState

        case hitByPitch :
            afterState = State()
            offencePlayerLst[0] = "nextStateHitter"
            afterState.setState(offensePlayerLst,
beforeState.getDefenseTeam().getDefenseTeamLst(), 0,
beforeState.getOut(), 0, beforeState.getTurn(),
beforeState.getIsHome())
            return afterState

        case baseOnBalls :
            afterState = State()
            offencePlayerLst[0] = "nextStateHitter"
            afterState.setState(offensePlayerLst,
beforeState.getDefenseTeam().getDefenseTeamLst(), 0,
beforeState.getOut(), 0, beforeState.getTurn(),
beforeState.getIsHome())
            return afterState

        case safe :
            afterState = State()
            offencePlayerLst[0] =
beforeState.getOffenseTeam().getOffenseTeamLst()[0]
            afterState.setState(offensePlayerLst,
beforeState.getDefenseTeam().getDefenseTeamLst(),
afterState.getBall(), afterState.getOut(), afterState.getStrike(),
afterState.getTurn(), afterState.getIsHome())
            return afterState

        case caughtStealing :
            afterState = State()
            offencePlayerLst[0] =
beforeState.getOffenseTeam().getOffenseTeamLst()[0]
            afterState.setState(offensePlayerLst,
beforeState.getDefenseTeam().getDefenseTeamLst(),
afterState.getBall(), afterState.getOut() + 1,
afterState.getStrike(), afterState.getTurn(),
afterState.getIsHome())
            return afterState

        case doubleSteal :
            afterState = State()
            offencePlayerLst[0] =
beforeState.getOffenseTeam().getOffenseTeamLst()[0]
            afterState.setState(offensePlayerLst,
beforeState.getDefenseTeam().getDefenseTeamLst(),
afterState.getBall(), afterState.getOut(), afterState.getStrike(),
afterState.getTurn(), afterState.getIsHome())
            return afterState

        case balk :
            afterState = State()
            offencePlayerLst[0] =
beforeState.getOffenseTeam().getOffenseTeamLst()[0]
            afterState.setState(offensePlayerLst,
beforeState.getDefenseTeam().getDefenseTeamLst(), 0,
afterState.getOut(), 0, afterState.getTurn(),
afterState.getIsHome())
            return afterState

        case pinchHitter :
            afterState = State()
            offencePlayerLst[0] = event.getPinchHitter()
            afterState.setState(offensePlayerLst,
beforeState.getDefenseTeam().getDefenseTeamLst(),
afterState.getBall(), afterState.getOut(), afterState.getStrike(),
afterState.getTurn(), afterState.getIsHome())
            return afterState

        case pinchRunner :
            afterState = State()
            offencePlayerLst[0] =
beforeState.getOffenseTeam().getOffenseTeamLst()[0]
            offencePlayerLst[event.getPinchRunnerBase()] =
event.getPinchRunner()
            afterState.setState(offensePlayerLst,
beforeState.getDefenseTeam().getDefenseTeamLst(),
afterState.getBall(), afterState.getOut(), afterState.getStrike(),
afterState.getTurn(), afterState.getIsHome())
```

```
            return afterState

        case pitcherChange :
            afterState = State()
            DefensePlayerLst = 
beforeState.getDefenseTeam().getDefenseTeamLst()
            DefensePlayerLst[0] = event.getBullpen()

afterState.setState(beforeState.getOffenseTeam().getOffenseTeamLst()
, DefensePlayerLst, beforeState.getBall(), beforeState.getOut(),
beforeState.getStrike(), beforeState.getTurn(),
beforeState.getIsHome())
            return afterState

         case fielderChange :
            afterState = State()
            DefensePlayerLst = 
beforeState.getDefenseTeam().getDefenseTeamLst()
            DefensePlayerLst[event.getChangePosition()] = 
event.getNewFielder()

afterState.setState(beforeState.getOffenseTeam().getOffenseTeamLst()
, DefensePlayerLst, beforeState.getBall(), beforeState.getOut(),
beforeState.getStrike(), beforeState.getTurn(),
beforeState.getIsHome())
            return afterState

        case benchClear :
            #identity function
            afterState = State()

afterState.setState(beforeState.getOffenseTeam().getOffenseTeamLst()
, beforeState.getDefenseTeam().getDefenseTeamLst(),
beforeState.getBall(), beforeState.getOut(),
beforeState.getStrike(), beforeState.getTurn(),
beforeState.getIsHome())
            return afterState

        case penalty :
            #퇴장의 경우 투수 퇴장과 타자 퇴장을 따로 다루어 주어야할것 같다. 나눈다
면 defenseteam과 offenseteam을 따로 바꿔주면 되므로 쉽다.

        case TeamChange :
            #저장되어있던 OffenseTeam과 defenseTeam을 불러오는것으로 해야함.
            #근데 어떻게 저장할지 아직 모르겠음. 일단 저장만 되면 매우 쉬움.
            #지금 생각에는 공수 교대때마다 global variable로 offenseteam과
defenseteam을 갱신하는 방식이 어떨까 싶음.
```

```
case benchChange :
    #무엇인지 모른다고 함...ㅠㅠ

case hitInference :
    #아직 구현하지 못했음
```