Se desea diseñar un sistema para manejar el ranking de posiciones de un torneo deportivo. En el torneo hay un conjunto fijo de equipos que juegan partidos (posiblemente más de un partido entre cada pareja de equipos) y el ganador de cada partido consigue un punto. Para el ranking, se decidió que entre los equipos con igual cantidad de puntos no se desempata, sino que todos reciben la mejor posición posible para ese puntaje. Por ejemplo, si los puntajes son: A: 5 puntos, B: 5 puntos, C: 4 puntos, D: 3 puntos, E: 3 puntos, las posiciones son: A: 1ro, B: 1ro, C: 3ro, D: 4to, E: 4to.

El siguiente TAD es una especificación para este problema.

```
Equipo es int
Partido es struct<ganador: Equipo, perdedor: Equipo>

TAD Torneo {
    obs equipos: conj<Equipo>
    obs partidos: seq<Partido>

    proc nuevoTorneo(): Torneo
        asegura res.equipos == {}
        asegura res.partidos == []

    proc registrarPartido(inout t: Torneo, in ganador: Equipo, in perdedor: Equipo)
        asegura t.partidos == old(t).partidos + [<ganador: ganador, perdedor: perdedor>]
        asegura t.equipos == old(t).equipos + {ganador, perdedor}

    proc posicion(in t: Torneo, in e: Equipo): int
        requiere e in t.equipos
        asegura res == cantConMasPuntos(t, puntosDe(t, e)) + 1

    proc puntos(in t: Torneo, in e: Equipo): int
        requiere e in t.equipos
        asegura res == t.puntosDe(e)

    proc masPuntos(in t: Torneo): Conjunto<Equipo>
        asegura forall e: Equipo :: e in res <==> t.cantConMasPuntos(puntosDe(t, e)) == 0

    aux puntosDe(t: Torneo, e: Equipo): int =
        sum i: int :: if 0 <= i < t.partidos && t.partidos[i].ganador == e then 1 else 0 fi

    aux cantConMasPuntos(t: Torneo, p: int): int =
        sum e: Equipo :: if e in t.equipos && puntosDe(t, e) > p then 1 else 0 fi

}
```

Se desea diseñar el sistema propuesto, teniendo en cuenta que las operaciones `puntos`, `registrarPartido` y `posicion` deben realizarse en $O(\log n)$, donde n es la cantidad de equipos registrados.

1. Describir la estructura a utilizar.
2. Escribir un pseudocódigo del algoritmo para las operaciones con requerimientos de complejidad.
3. Se desea que, además de saber quién perdió y ganó, al registrar un partido se sepa cuántos goles hizo cada equipo. Con esa información, se desea contar con el proc `maxDiferencia`, que obtenga el equipo que tiene mejor diferencia de gol actualmente. Si hay varios, deberá devolver alguno de ellos, sin importar cuál.
   a) Realizar los cambios en la estructura para contemplar esta modificación.
   b) Modificar los algoritmos anteriores, de ser necesario, y justificar su cota temporal.