# DIE HARD WITH A VENGEANCE

## (Y el Problema de los Bidones...)

McClane tiene mucha resaca y está siendo atormentado por un tal Simon (*Saimon*) que lo obliga a moverse por todo New York invitándolo a jugar al famoso juego de “Simon dice”, bajo la amenaza de hacer explotar bombas en caso de no obedecer. Llamó al S.T.O.P (Spastic Team Of Programmers) para que lo ayuden a resolver el problema.

Las órdenes que da este malvado señor Simon las tenemos representadas por un predicado ***simonDice/2*** que relaciona una “orden” con una lista de personas involucradas que terminan realizando dicha orden.

```
% irA(lugar, barrio)) /  irA(calle1, calle2, barrio)) / hacer(accion, barrio)
simonDice(irA(calle138, amsterdam, harlem), [mcClane]).
simonDice(hacer(vestirCartel, harlem), [mcClane]).
simonDice(irA(west72nd, broadway, upperWestSide), [mcClane, zeus]).
simonDice(hacer(resolverAcertijo, upperWestSide), [mcClane, zeus]).
simonDice(irA(wallStreetStation, worldTradeCenter), [mcClane, zeus]).
simonDice(hacer(atenderTelefono, worldTradeCenter), [zeus]).
simonDice(irA(tompkinsSquarePark, eastVillage), [mcClane, zeus]).
simonDice(hacer(desarmarBomba, eastVillage), [mcClane, zeus]).
simonDice(irA(yankeeStadium, bronx), [zeus]).
```

Y también contamos con la información de todos los ataques que realiza Simon, separados de forma tal de identificar las amenazas de bomba, de las que realmente explotaron. Las amenazas se representan con el barrio y el tiempo que duraron.


- En Sixth Avenue hubo una explosión.
- En Chinatown solo hubo una amenaza que duró 5 horas.
- En Upper West Side la amenaza duró 2 horas.
- Hubo una explosión en World Trade Center y otra en un bote.
- En la escuela de Arthur hubo una amenaza durante 3 horas.

A partir de ésta información se pide resolver los siguientes requerimientos:

**Nota**: Tener en cuenta que todos los predicados deben ser inversibles salvo que se indique lo contrario.

1. Barrios y personas
    a. Relacionar a la persona y al barrio si esa persona estuvo en ese barrio.
    ```
    pasoPor(Persona, Barrio).
    Persona = mcClane
    Barrio = harlem
    ```
    b. Relacionar a una persona con todos los barrios por los que pasó, sin repetir.
2. Conocer:

   a. A un posible culpable de atentado en un barrio, que es una persona que estuvo en un barrio en el cual hubo una explosión.
   b. Si un barrio dado explotó por culpa de McClane, que se cumple cuando ese barrio explotó y McClane fue el único que estuvo ahí.

3. Quién es afortunado, que es aquel que estuvo en barrios amenazados por un total de al menos 8 horas (entre todos los barrios), pero en ninguno que haya explotado.

4. El jefe quiere poder hacer la siguiente consulta:
   `?- puedoEcharA(mcClane).`
   Y, si no lo puede echar, quiere poder volver a preguntar para saber a quién puede echar.
   El jefe puede echar a cualquiera que, en todos los barrios que estuvo, o bien hubo explosiones o bien el barrio estuvo amenazado por al menos 3 horas.

5. Ahora están en pleno Manhattan con una bomba en sus manos, la cual McClane activó sin querer, y un problema lógico que sirve para desactivarla. Sólo tiene el tiempo de medio parcial para resolverlo. El problema consiste en responder la siguiente pregunta: *¿Cómo obtener exactamente 4 litros de agua con un bidón de 5 litros y otro de 3 litros?*

Existen distintas acciones a realizar con cada bidón y las representamos de la siguiente manera:

```
accion(llenarChico).
accion(llenarGrande).
accion(vaciarChico).
accion(vaciarGrande).
accion(ponerChicoEnGrande).
accion(ponerGrandeEnChico).
```

Para resolver esto tener en cuenta:

   a. Un modelo el cual me permita manejar el estado de ambos bidones. El estado es la cantidad de litros de agua que tiene.
   b. Realizar un predicado que relacione una acción (o paso) con un estado de bidones anterior y un estado de bidones posterior a realizar dicha acción:
      i. Ejemplo 1: si tengo los bidones vacíos y la acción que debo realizar es llenar el bidón chico, lo que debe pasar es que el estado del bidón chico debe “cambiar”, mientras que el grande queda igual. Por otro lado, teniendo en cuenta el enunciado, no sería posible realizar esa misma acción si el bidón chico ya esta lleno
      ii. Si tengo el grande lleno y el chico vacío y la acción es de pasar del grande al chico, entonces el chico queda lleno, y el grande queda con el resto del agua que no cupo en el chico.
      - PRO-TIP: Existen operadores min y max. Por ejemplo: `N is min(4, 5)`
      - Este predicado solo requiere ser inversible para el estado posterior.
   c. Pensar un predicado que relacione un estado de bidones, con una cantidad de litros como **objetivo**. Dicho predicado debe ser verdadero cuando se alcanza cierto objetivo (el objetivo debe ser genérico, no necesariamente es 4 litros). Este predicado no requiere ser inversible.
   d. Implementar el predicado `resolverBidones/3` que resuelva el problema, teniendo en cuenta que siempre se comienza con los bidones vacíos. Ejemplo:
      ```
      ?- resolverBidones(4, 9, Acciones).
      Acciones = [llenarChico, ponerChicoEnGrande, llenarChico, ponerChicoEnGrande, vaciarGrande, ponerChicoEnGrande, llenarChico, ponerChicoEnGrande]
      ```
      En el ejemplo, 4 es la cantidad de litros a obtener en cualquiera de los 2 bidones, y 9 es un límite

máximo de acciones a realizar (hay 8 en nuestra lista de ejemplo).
Otro ejemplo válido:

```
?- resolverBidones(3, 1, Acciones)
Acciones = [llenarChico]
```

Este predicado solo requiere ser inversible para el tercer argumento.