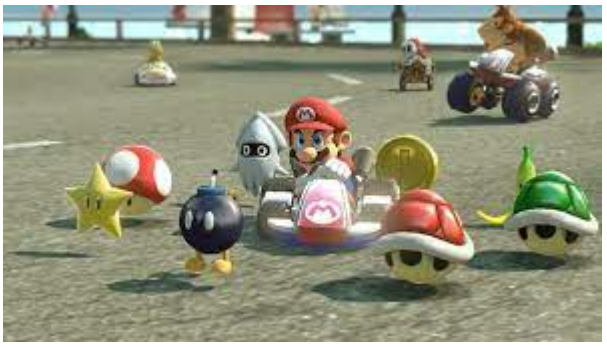
# Carreras



Queremos armar un programa que nos permita simular unas fantásticas carreras de autos en las cuales cada vehículo avanza tan rápido como puede para consagrarse campeón, aprovechando del uso de algunos poderes especiales (o power ups) que encuentren a lo largo del trayecto para sacar ventaja por sobre los demás autos.

De cada auto conocemos su **color** (un string que nos servirá para identificarlo durante el desarrollo de la carrera), la **velocidad** a la que está yendo y la **distancia** que recorrió, ambos valores de tipo entero.

De la carrera sólo nos interesa el estado actual de los autos que están participando, lo cual nos permitirá analizar cómo viene cada uno, y posteriormente procesar aquellos eventos que se den en la carrera para determinar el resultado de la misma.

Teniendo en cuenta lo descrito anteriormente se pide resolver los siguientes puntos **explicitando el tipo de cada función desarrollada** y **utilizando los conceptos aprendidos del Paradigma Funcional**, poniendo especial énfasis en el uso de **Composición**, **Aplicación Parcial** y **Orden Superior**.

1. Declarar los tipos **Auto** y **Carrera** como consideres convenientes para representar la información indicada y definir funciones para resolver los siguientes problemas:
    a. Saber si un auto **está cerca** de otro auto, que se cumple si son autos distintos y la distancia que hay entre ellos (en valor absoluto) es menor a 10.
    b. Saber si un auto **va tranquilo** en una carrera, que se cumple si no tiene ningún auto cerca y les va ganando a todos (por haber recorrido más distancia que los otros).
    c. Conocer en qué **puesto** está un auto en una carrera, que es 1 + la cantidad de autos de la carrera que le van ganando.

2. Desarrollar las funciones necesarias para manipular el estado de los autos para que sea posible:
    a. Hacer que un auto **corra** durante un determinado tiempo. Luego de correr la cantidad de tiempo indicada, la distancia recorrida por el auto debería ser equivalente a la distancia que llevaba recorrida + ese tiempo * la velocidad a la que estaba yendo.
    b.
        i. Dado un modificador de velocidad de tipo **Int -> Int**, queremos poder **alterar la velocidad** de un auto de modo que su velocidad final sea la resultante de usar dicho modificador con su velocidad actual.
        ii. Usar la función del punto anterior para **bajar la velocidad** de un auto en una cantidad indicada de modo que se le reste a la velocidad actual la cantidad indicada, y como mínimo quede en 0, ya que no es válido que un auto quede con velocidad negativa.

3. Como se explicó inicialmente sobre las carreras que queremos simular, los autos que participan pueden gatillar poderes especiales a los que denominamos **power ups**.
   Estos poderes son variados y tienen como objetivo impactar al estado general de la carrera, ya sea afectando al auto que lo gatilló y/o a sus contrincantes dependiendo de qué poder se trate.

   ***Nota:*** *disponemos de una función* ***afectarALosQueCumplen :: (a -> Bool) -> (a -> a) -> [a] -> [a]*** *que puede ser de utilidad para manipular el estado de la carrera. Ver pág. 2 para más detalles.*

   Inicialmente queremos poder representar los siguientes power ups, pero debería ser fácil incorporar más power ups a futuro para enriquecer nuestro programa:
    a. **terremoto**: luego de usar este poder, los autos que están cerca del que gatilló el power up bajan su velocidad en 50.

b. **miguelitos**: este poder debe permitir configurarse con una cantidad que indica en cuánto deberán bajar la velocidad los autos que se vean afectados por su uso. Los autos a afectar son aquellos a los cuales el auto que gatilló el power up les vaya ganando.
c. **jet pack**: este poder debe afectar, dentro de la carrera, solamente al auto que gatilló el poder. El jet pack tiene un impacto que dura una cantidad limitada de tiempo, el cual se espera poder configurar.
   Cuando se activa el poder del jet pack, el auto afectado duplica su velocidad actual, luego corre durante el tiempo indicado y finalmente su velocidad vuelve al valor que tenía antes de que se active el poder.
   Por simplicidad, no se espera que los demás autos que participan de la carrera también avancen en ese tiempo.

Como se mencionó anteriormente, disponemos de la siguiente función para usar dentro de la resolución:

```
afectarALosQueCumplen :: (a -> Bool) -> (a -> a) -> [a] -> [a]
afectarALosQueCumplen criterio efecto lista
        = (map efecto . filter criterio) lista ++ filter (not.criterio) lista
```

4. A partir de todo lo construido hasta ahora queremos finalmente simular una carrera, para lo cual se provee una lista de **eventos**, que son funciones de tipo **Carrera -> Carrera** que permiten ir de un estado de la carrera al siguiente, y el estado inicial de la carrera a partir del cual se producen dichos eventos.

   Con esta información buscamos generar una **tabla de posiciones**, que incluye la información de en qué **puesto** quedó cada auto asociado al **color** del auto en cuestión.

   Se pide:
   a. Desarrollar la función: **simularCarrera :: Carrera -> [Evento] -> TablaDePosiciones**
      que permita obtener la tabla de posiciones a partir del estado final de la carrera, el cual se obtiene produciendo cada evento uno detrás del otro, partiendo del estado de la carrera recibido. La tabla de posiciones debería ordenarse en base a la posición final de los autos.

      Ya disponemos de una función `ordenarPor :: Ord a => (b -> a) -> [b] -> [b]` que puede usarse para ordenar la tabla de posiciones.
   b. Desarrollar las siguientes funciones de modo que puedan usarse para **generar los eventos** que se dan en una carrera:
      i. **correnTodos** que hace que todos los autos que están participando de la carrera corran durante un tiempo indicado.
      ii. **usaPowerUp** que a partir de un power up y el color del auto que gatilló el poder en cuestión, encuentre el auto correspondiente dentro del estado actual de la carrera para que use el power up y produzca los efectos esperados de su uso.
   c. Probar lo desarrollado simulando una carrera con autos de colores rojo, blanco, azul y negro que vayan inicialmente a velocidad 120 y su distancia recorrida sea 0, de modo que ocurran los siguientes eventos en el orden en el que se indican:
      - todos los autos corren durante 30 segundos
      - el azul usa el power up de jet pack por 3 segundos
      - el blanco usa el power up de terremoto
      - todos los autos corren durante 40 segundos
      - el blanco usa el power up de miguelitos que reducen la velocidad en 20
      - el negro usa el power up de jet pack por 6 segundos
      - todos los autos corren durante 10 segundos

      Se espera las posiciones de cada auto luego de los eventos indicados sean:
      *1. "azul"  2. "blanco"  3. "negro"  4. "rojo"*
5. Se quiere agregar un nuevo power up: un **misil teledirigido**. Para poder activarlo se debe indicar el color del auto al que se quiere impactar, y si la velocidad de dicho auto a afectar es menor a 50, luego del

impacto del misil debería quedar con velocidad igual a 10. Además, debería incrementar la distancia recorrida del auto impactado en 5 unidades, pero sólo si el mismo le va ganando al auto que usó el power up (como consecuencia de ser golpeado por el misil desde atrás).

a. ¿La solución desarrollada hasta este punto permite agregar el nuevo power up o sería necesario cambiar algo de lo desarrollado en los puntos anteriores? Justificar.
b. Implementar el misil teledirigido.