Este trabajo consiste en implementar funciones que permitan comprimir y descomprimir un video, usando la técnica de *differencing*. Un video está compuesto por *frames*: cada frame es una matriz de pixels representada como una lista de listas en donde cada lista interna representa una fila. A su vez, estas filas contienen ternas RGB representando cada pixel.

En esta técnica de compresión, cada frame es comparado con su predecesor. Si la diferencia entre ellos es chica (sólo unos pocos pixels), el nuevo frame es comprimido: por cada pixel que haya cambiado se guardan sus coordenadas (fila y columna en el frame) y la diferencia entre los valores del pixel en ambos frames. En cambio, si la diferencia entre el frame y su predecesor es muy grande (muchos pixels cambiaron), el nuevo frame se guarda completo, sin comprimir.

A la hora de descomprimir el video, el valor de cada pixel se obtendrá del valor del mismo en el frame descomprimido anterior, y se le suma, de haberla, la diferencia guardada en la compresión.

Código base

Junto con este enunciado deberán bajar el archivo **tp-2015-2C.hs** que contiene un esqueleto del trabajo para que utilicen como base. En este archivo encontrarán:

- La definición de los tipos que utilizaremos.
- Los encabezados de las funciones que deben implementar (a veces con algún ejemplo).
- Funciones brindadas por la cátedra (para el último ejercicio).
- Videos de ejemplo para que puedan hacer pruebas. Recomendamos fuertemente agregar más ejemplos propios, distintos a los de la cátedra, para entender mejor los problemas.

Ejercicios

1. Implementar la función `ultimoFrame :: Video -> Frame`
   que devuelve el último frame del video. Por último frame debe entenderse el más recientemente agregado, o en caso de no haberlo, el frame inicial.

2. Implementar `norma :: (Integer, Integer, Integer) -> Float`
   que devuelve la norma 2 del vector representado por una terna de 3 componentes. Para este ejercicio es posible que deban utilizar la función `fromInteger` para convertir de `Integer` a `Float`. $norma_2(< x_1, x_2, x_3 >) = \sqrt{x_1^2 + x_2^2 + x_3^2}$

3. Al comparar dos frames, se puede obtener un `FrameComprimido`. Este frame se representa como una lista de valores que contiene sólo las posiciones de los pixels que cambiaron significativamente, junto con el valor del cambio. Se pide programar la función `pixelsDiferentesEnFrame :: Frame -> Frame -> Float -> FrameComprimido` que devuelve el frame comprimido que resulta de hallar los pixels diferentes entre los dos frames (de igual tamaño) recibidos como parámetro. Dos pixels se consideran diferentes si la norma 2 de la diferencia entre los mismos es mayor a cierto umbral $u \in \mathbb{R}$ (indicado en el tercer parámetro). **Aclaración**: Los índices de fila y columna en el resultado deberán comenzar desde 0. Por ejemplo, el valor (1, 2, (-2, 0, 1)) en el resultado representa que en la segunda fila, tercer columna, la diferencia entre el valor de los pixeles del primer y segundo frame es (-2, 0, 1).

4. Un `VideoComprimido` está compuesto por frames normales y frames comprimidos. Para la construcción de videos comprimidos se lleva a cabo el siguiente procedimiento:

a) Se inicia con el primer frame del video original.

b) En caso de haber 2 frames consecutivos muy similares se agrega al video comprimido el `FrameComprimido` que contiene el cambio para cada pixel. Sí no, se agrega el frame sin comprimir. Para determinar si dos frames son similares se utilizará un umbral $n \in \mathbb{N}$ que indica la cantidad máxima de pixels distintos entre los dos frames. Recordar que para determinar pixels distintos, existe el umbral $u \in \mathbb{R}$ utilizado en el punto anterior.

Implementar entonces la función
`comprimir :: Video -> Float -> Integer -> VideoComprimido`
que dado un video v, un umbral para la comparación de pixels u y un umbral para la cantidad de pixels diferentes n, aplica la técnica de differencing para comprimir el video. Para este ejercicio es posible que deban utilizar la función `fromIntegral` para convertir de `Int` a `Integer`.

5. Implementar la función `descomprimir :: VideoComprimido -> Video`
que dado un video v comprimido usando la técnica de differencing, lo descomprime. Para esta función puden utilizar las siguientes funciones provistas por la cátedra.
Aclaración: estas funciones auxiliares pueden o no ser necesarias dependiendo de su solución.

   - `aplicarCambio :: Frame -> FrameComprimido -> Frame` que devuelve el resultado de aplicar los cambios especificados por el frame comprimido al frame original (ver ejemplo en el código).
   - `sumarCambios :: FrameComprimido -> FrameComprimido -> FrameComprimido` que une los cambios aplicados por dos frames comprimidos (ver ejemplo en el código).

Condiciones de entrega:

- El trabajo práctico se debe hacer en grupos de **exactamente 3 personas** (salvo cuando lo contrario sea inexorable y con expresa autorización de la cátedra).
- La composición de cada grupo se debe comunicar a Silvina Dengra (`sdengra@dc.uba.ar`).
- Se debe entregar el código por mail a la lista de docentes (`algebra1-doc@dc.uba.ar`) a más tardar **antes del comienzo de la clase** del turno del día de entrega.
- Todos los integrantes deben ser **del mismo turno**. Recomendamos fuertemente que así sea. En caso de querer pedir una excepción a esta norma, debe hacerse con antelación y ateniéndose a todas las consecuencias (por ejemplo, la fecha de entrega será la más temprana, no la más tardía de las alternativas; ídem con la fecha de coloquio, donde deberán estar todos presentes aunque alguien curse en otro turno).
- Las funciones pedidas deben llamarse igual y deben tener la misma signatura que la que se pide. Pueden agregarse funciones auxiliares: cada una debe llevar un comentario explicando qué hace, qué es cada parámetro y qué restricciones tienen (de haberlas).