**TECNOLÓGICO DE COSTA RICA I SEMESTRE 2020**

**INGENIERÍA EN COMPUTACIÓN**

**IC-4700 LENGUAJES DE PROGRAMACIÓN**

**TAREA PROGRAMADA 1**

En esta tarea se debe implementar una aplicación en Haskell que alinee un texto para una salida en una terminal con texto de tamaño fijo. El alineamiento se obtiene insertando espacios en blanco y dividiendo palabras. Por ejemplo, dado el texto:

**texto = "Aquel que controla el pasado controla el futuro. Aquel que controla el presente controla el pasado."**

se quiere separar en líneas de 17 columnas:

**12345678901234567**

**Aquel que contro-**

**la el pasado con-**

**trola el futuro.**

**Aquel que contro-**

**la el presente**

**controla el pasa-**

**do.**

Este es un problema más intrincado de lo que parece. Se obtendrá la solución paso por paso, definiendo una serie de pequeñas y bien definidas funciones conforme se avanza. Esto es considerado como el estilo correcto de programación funcional.

Primero se representarán las tiras de texto como una lista de tokens que será conocida como **Line**. Los tokens pueden ser palabras, palabras separadas o espacios en blancos insertados.

**type Line = [Token]**

**data Token = Word String | Blank | HypWord String**

**deriving (Eq,Show)**

1. Defina una función que convierta una tira de entrada en una Line. Puede asumir que la entrada no contiene palabras separadas.

**string2line :: String -> Line**

**string2line text ⇒ [Word "Aquel",Word "que",Word "controla", ...]**

1. Defina una función que convierta de vuelta un Line en una tira. Se supone que hay un espacio en blanco luego de cada palabra menos la última.

**line2string :: Line -> String**

**line2string [Word "Aquel",Word "que"] ⇒ "Aquel que"**

Si el texto original solo tiene un espacio entre cada palabra entonces se da:

**line2string (string2line text) == text ⇒ True"**

Si el **Line** contiene palabras separadas, estas se deben mostrar poniendo un guion al final. Por ejemplo:

**line2string [Word "Aquel",Word "que",HypWord "contro",Word "la"]**

**⇒ "Aquel que contro- la"**

Si aparece un Blank explícito se toma como un espacio en blanco adicional. Se ignoran los Blank al inicio o al final. En el siguiente ejemplo note que entre “Aquel” y “que” hay 3 espacios en blanco, 1 para separar las palabras y 2 más por los Blank:

**line2string [Blank, Word "Aquel",Blank, Blank,Word "que", Blank]**

**⇒ "Aquel que"**

1. Defina una función que calcule la longitud de un token:

**tokenLength :: Token -> Int**

**tokenLength (Word "Aquel") ⇒ 5**

**tokenLength (HypWord "contro") ⇒ 7** -- *Tomar en cuenta el "-"*

**tokenLength (Blank) ⇒ 1**

1. Defina una función que calcule la longitud de un **Line**:

**lineLength [Word "Aquel",Word "que",HypWord "contro",Word "la"]⇒ 20**

**lineLength [Word "Aquel",Word "que",HypWord "con"] ⇒ 14**

1. Defina un función que parta una línea de modo que no sea más larga que una longitud dada. La función retorna un par de líneas: la primera es la línea con la longitud limitada, y la segunda es su continuación.

**breakLine :: Int -> Line -> (Line,Line)**

**breakLine 1 [Word "Aquel",Word "que",Word "controla"]  
 ⇒ ([],[Word "Aquel",Word "que",Word "controla"])**

**breakLine 4 [Word "Aquel",Word "que",Word "controla"]**

**⇒ ([],[Word "Aquel",Word "que",Word "controla"])**

**breakLine 5 [Word "Aquel",Word "que",Word "controla"]**

**⇒ ([Word "Aquel"],[Word "que",Word "controla"]**

**breakLine 9 [Word "Aquel",Word "que",Word "controla"]**

**⇒ ([Word "Aquel",Word "que"],[Word "controla"])**

**breakLine 100 [Word "Aquel",Word "que",Word "controla"]**

**⇒ ([Word "Aquel",Word "que",Word "controla"],[])**

**breakLine 3 [] ⇒ ([],[])**

1. Defina una función auxiliar **mergers** que tome una lista de tiras y genere todas las formas de concatenarlas en orden dejando sin concatenar dos tiras acumuladas:

**mergers ["co","nt","ro","la"]**

**⇒ [("co","ntrola"),("cont","rola"),("contro","la")]**

**mergers ["co","nt"] ⇒ [("co","nt")]**

**mergers ["co"] ⇒ []**

1. Para poder alinear las líneas de forma agradable, debe ser posible separar palabras largas. Aunque es posible implementar las reglas de separación en sílabas del español, por el momento se tomará un camino más simple y se supondrá que se dispone de una lista de palabras con su separación correcta. Esto se implementará usando el módulo Data.Map:

**type HypMap = Data.Map.Map String [String]**

Por ejemplo

**enHyp :: HypMap**

**enHyp = Data.Map.fromList [ ("controla",["con","tro","la"]),**

**("futuro",["fu","tu","ro"]),**

**("presente",["pre","sen","te"])]**

Definir una función de separación de palabras que separe un token **Word** de todas las formas posibles que se pueden derivar del **Data.Map** de arriba. Aproveche la función **mergers** solicitada antes.

**hyphenate :: HypMap -> Token -> [(Token,Token)]**

**hyphenate enHyp (Word "controla")**

**⇒ [(HypWord "con",Word "trola"),(HypWord "contro",Word "la")]**

**hyphenate enHyp (Word "firefox")**

**⇒ [] -- no está en el map**

Nota: si una palabra tiene puntuación adicionada al final de la misma, se debe quitar la puntuación antes de buscar en el map; luego de separada la palabra, se debe restaurar la puntuación en la segunda parte de la palabra.

**hyphenate enHyp (Word "futuro.")**

**⇒ [(HypWord "fu",Word "turo."),(HypWord "futu",Word "ro.")] hyphenate enHyp (Word "futuro...")**

**⇒ [(HypWord "fu",Word "turo..."),(HypWord "futu",Word "ro...")]**

1. Definir ahora una función que encuentre las diferentes formas en que se puede separar una línea usando las diferentes formas en que la última palabra de la línea se pueda separar. Adicionalmente, la línea separada al inicio no puede sobrepasar una longitud dada.

**lineBreaks :: HypMap -> Int -> Line -> [(Line,Line)]**

**lineBreaks enHyp 17 [Word "Aquel",Word "que",Word "controla"]**

**⇒ [([Word "Aquel",Word "que"], [Word "controla"]),**

**([Word "Aquel",Word "que",HypWord "con"], [Word "trola"]),**

**([Word "Aquel",Word "que",HypWord "contro"], [Word "la"])]**

**lineBreaks enHyp 12 [Word "Aquel"] ⇒ [([Word "Aquel"],[])]**

1. Definir una función **insertBlanks** que distribuya un número dado de espacios en blanco entre las palabras. La distribución se hará colocando un Blank en posiciones consecutivas de izquierda a derecha. Esto se repite de nuevo de izquierda a derecha hasta que se haya colocado la cantidad de espacios en blanco pedidos.

**insertBlanks** **2 [] ⇒ []**

**insertBlanks** **2 [Word "hola"] ⇒ [Word "hola"]**

**insertBlanks** **2 [Word "hola", Word "mundo", Word "cruel"]**

**⇒ [Word "hola", Blank, Word "mundo", Blank, Word "cruel"]**

**insertBlanks** **3 [Word "hola", Word "mundo", Word "cruel"]**

**⇒ [Word "hola", Blank,Blank, Word "mundo", Blank, Word "cruel"]**

**insertBlanks** **5 [Word "hola", Word "mundo", Word "cruel"]**

**⇒ [Word "hola", Blank,Blank,Blank, Word "mundo", Blank,Blank,**

**Word "cruel"]**

**insertBlanks** **5**

**[Word "hola", Word "mundo", Word "cruel", Word "adios"]**

**⇒ [Word "hola",Blank,Blank, Word "mundo", Blank,Blank,   
 Word "cruel", Blank, Word "adios"]**

1. Finalmente, definir una función **separarYalinear** que reciba una tira de caracteres y un tamaño de línea, y que devuelva una lista de tiras de caracteres que no sean más largas que el tamaño especificado La función también recibe dos banderas. La primera indica si debe usar separación de palabras o no: el valor de NOSEPARAR indica que las líneas se crean sin separar palabras; el valor de SEPARAR indica que para cada línea se debe intentar separar la palabra que no cabe, usando la separación que incluya la mayor parte de la palabra en la línea anterior. La segunda bandera indica si se deben ajustar las líneas, esto es, se deben insertar espacios en blanco para que las líneas sean exactamente del tamaño dado; con excepción de la última línea, a la cual nunca se le inserta un espacio en blanco. Para esta segunda bandera, el valor de NOAJUSTAR indica que no deben insertarse espacios en blanco para alinear el texto; si el valor es AJUSTAR entonces se insertan espacios en blanco en las líneas.

**separarYalinear 20 NOSEPARAR NOAJUSTAR** **"** **Quien controla el pasado controla el futuro. Quien controla el presente controla el pasado."**

**⇒**

**-- 12345678901234567890**

**[ "Quien controla el",**

**"pasado controla el",**

**"futuro. Quien",**

**"controla el presente",**

**"controla el pasado."]**

**separarYalinear 20 NOSEPARAR AJUSTAR** **"** **Quien controla el pasado controla el futuro. Quien controla el presente controla el pasado."**

**⇒**

**-- 12345678901234567890**

**[ "Quien controla el",**

**"pasado controla el",**

**"futuro. Quien",**

**"controla el presente",**

**"controla el pasado."]**

**separarYalinear 20 SEPARAR NOAJUSTAR** **"** **Quien controla el pasado controla el futuro. Quien controla el presente controla el pasado."**

**⇒**

**-- 12345678901234567890**

**[ "Quien controla el",**

**"pasado controla el",**

**"futuro. Quien con-",**

**"trola el presente",**

**"controla el pasado."]**

**separarYalinear 20 SEPARAR AJUSTAR** **"** **Quien controla el pasado controla el futuro. Quien controla el presente controla el pasado."**

**⇒**

**-- 12345678901234567890**

**[ "Quien controla el",**

**"pasado controla el",**

**"futuro. Quien con-",**

**"trola el presente",**

**"controla el pasado."]**

**Aplicación**

Elaborar una aplicación de Haskell que permita realizar operaciones como las anteriores en un ciclo iterativo. El programa **conteo-palabras.hs** que se les adjunta, muestra cómo Haskell puede implementar un ciclo iterativo y guardar y actualizar un estado. A continuación se describen las operaciones que debe implementar.

* **load** *archivo*   
  Carga la separación de palabras almacenada en el ***archivo*** cuya ruta se especifica. El sistema almacena un único diccionario de separaciones de palabras. Inicialmente ese diccionario es nulo. Si este comando se ejecuta varias veces, el sistema simplemente elimina el diccionario anterior y carga el nuevo. El formato del ***archivo*** es como se muestra en el archivo dic1.txt que se adjunta. El comando debe dar la cantidad de palabras cargadas.

**>> load dic1.txt***Diccionario cargado (25 palabras)*

* **show**  
  Imprime el diccionario actualmente almacenado

**>> show**[("razón",["ra","zón"]),("sinrazón",["sin","ra","zón"]) …]

* **ins** *palabra palabra-separada*  
  Incluye una nueva palabra en el diccionario almacenado

**>> ins asuntos a-sun-tos**

*Palabra asuntos agregada*

* **save** *archivo*   
  Guarda el diccionario almacenado en memoria en el *archivo* cuya ruta se especifica. Se debe guardar siguiendo el formato del diccionario de palabras que se muestra en el archivo dic1.txt que se adjunta. El comando debe dar la cantidad de palabras guardadas.

**>> save dic2.txt***Diccionario guardado (26 palabras)*

* **split** *longitud separar? ajustar? texto*  
  Tomar el *texto* (que es el resto de la línea de comando) y aplicar la función ***separarYalinear*** . Los valores de los parámetros *separa?* y *ajustar?* son ‘s’ o ‘n’. Se debe mostrar el resultado formateando la salida.

**>> split 20 n s Quien controla …  
Quien controla el**

**pasado controla el**

**futuro. Quien con-**

**trola el presente**

**controla el pasado.**

* **splitf** *longitud separar? ajustar? archivo1 archivo2*  
  Es como el comando anterior (split**)**, solo que el texto lo lee de *archivo1*. El parámetro archivo2 es opcional. Si se incluye el parámetro *archivo2*, el resultado es guardado en la ruta especificada por dicho parámetro.

**>> splitf 20 n s prueba1.txt  
Quien controla el**

**pasado controla el**

**…**

* **exit**Salir del ciclo iterativo y terminar la ejecución.

**Pruebas**

Aplicar la función separarYalinear con map al siguiente texto usando las opciones que se indican

**>> load dic1.txt**

**>> split 20 n n Quien controla el pasado controla el futuro. Quien controla el presente controla el pasado.**

**>> split 20 n s Quien controla el pasado controla el futuro. Quien controla el presente controla el pasado.**

**>> split 20 s n Quien controla el pasado controla el futuro. Quien controla el presente controla el pasado.**

**>> split 20 s s Quien controla el pasado controla el futuro. Quien controla el presente controla el pasado.**

**>> ins elementalmente e-le-men-tal-men-te**

**>> split 20 n n texto1.txt**

**>> split 20 s s texto1.txt**

**>> split 20 n n texto1.txt salida1.txt**

**>> split 20 s s texto1.txt salida1.txt**

**>> split 20 n n texto2.txt salida2.txt**

**>> split 20 s s texto2.txt salida2.txt**

**>> ins elementalmente e-le-men-tal-men-te**

**>> split 24 s n Quien controla elementalmente el pasado controla el futuro. Quien controla el presente controla el pasado.**

**>> save dic1b.txt**

**>> fin**