Sí explicaremos la solución dada en Haskell. La función se llama qsort y toma una lista como entrada. Debemos tener en cuenta para interpretar el código anterior que en Haskell las funciones definidias por el usuario tienen más precedencia que cualquier otra cosa (salvo que se diga la contrario), por lo que f 5 \* 2 = (f 5) \* 2. La función se ha definido por lo que se llama ajuste de patrones (pattern matching), lo que significa que el sistema probará el argumento pasado a la función con los patrones de arriba a abajo, y aplicará el primer patrón que se ajuste a los datos recibidos. Por ello, la primera definición se ajusta a [], que en Haskell es la lista vacía (una lista formada por 1, 2 y 3 sería [1,2,3] así que tiene sentido que la lista vacía sean sólo dos corchetes), y cuya ordenación de devolver la lista vacía. El segundo patrón de definición se ajusta a una lista con al menos un elemento (el operador (:) simplemente pone un elemento en la cabeza de una lista, por lo que 0 : [1,2,3] = [0,1,2,3]), el patrón (x:xs) se cumple cuando se recibe una lista con cabeza x y cola xs (que debe ser una lista, ya sea vacía o no), por lo que representa una lista que tiene, al menos, un elemento. Como necesitaremos usar la cabeza de la lista después, podemos extraerla muy elegantemente mediante la expresión del patrón usado (aunque también podríamos haber usado como patrón una sola variable, que representaría la lista completa, y luego usar la función head para obtener la cabeza de la lista, pero es mucho menos elegante ya que se identificaría con cualquier dato recibido, lista o no). En este caso, si tenemos una lista no vacía que empieza por un elemento, x, la lista ordenada completa se consigue ordenando de forma recursiva, por una parte, todos los elementos menores a x y poniéndolos delante de x, y por otra parte, ordenando todos los elementos mayores a x y poniéndolos después de la aparición de x. Para conseguir la concatenación de estas listas ya ordenadas usamos el operador ++. Como el elemento x no es una lista, el operador ++ no funciona sobre él, por lo que lo convertimos en lista unitaria poniéndolo entre corchetes (es sencillo, [x] es la lista formada por x). ¿Cómo se calculan las listas menores y mayores? Debido a que no importa la secuenciación en Haskell, se definen debajo de la función usando la notación where (que permite usar los parámetros que recibe la función que estamos definiendo en sus definiciones). Por lo demás, se parece mucho a las funciones definidas al principio de esta entrada, por lo que no daremos más detalles.