Converts a long to a series of symbols - the encoding of the number in any base, given any symbol system. Return a String which is the encoding of the long in the symbol system given. The list of symbols will be given as an array of characters; the base of the encoding is simply the number of symbols in the array. For instance, if the symbols given are the array {'0','1','2','3'}, then this is a base-4 encoding and the symbols are the ordinary digits. Thus, the encoding of the number 20 (decimal) would be "110" (4^2 + 4^1). If the array was {'0','1'}, then this is base-2, and the encoding would be "10100" (2^4 + 2^2). But since the symbols are given in an array, don't expect that the encoding will necessarily look like a number. If the array given is {'$','.'}, then 20 (decimal) would be encoded as ".$.$$". (This is the same encoding as the previous example - just with different symbols.) Why the complexity? So that it's impossible to use standard Java library functions to format a number! (Plus, it's cool to see that the symbols don't really matter!) Note about hard-coding: It is perfectly OK to hardcode for the value 0 in this method. That's what a professional would do.

ERROR HANDLING: - Throw IllegalArgumentException if the number to encode is negative. - Throw IllegalArgumentException if the length of the array is less than 2.

SYNTAX TRICKS: - The index to an array must be an int, not a long. So when you calculate an index into the symbols[] array, make sure that you cast the index to an int \*AFTER\* you have calculated it. Otherwise, your code won't compile. You can convert a single char to a String by concatenating it to the empty string.

HINT: Use recursion to make this method easier. Figure out one symbol - and then use recursion to find the rest of the encoding.

@param val A non-negative number

@param symbols The symbols to encode the number

@return A String giving the encoding