Homework 4

CSC 121-2
Fall 2019

1. Define a function `repeats :: Eq a => [a] -> Bool` so that `repeats xs` is true if and only if `xs` has two equal elements next to each other.

2. Represent a polynomial using a list of its coefficients, starting with the constant coefficient and going only as high as necessary. For example `[1,5,3]` represents the polynomial $3x^2 + 5x + 1$. Write a recursive function `eval` that takes a polynomial represented this way and a value for x and returns the value of the polynomial at the given x. For example, `eval [1,5,3] 2` should give 23.

3. Make a new version of `qsort`, but this version should be a curried function that acts on a curried function that determines the order of two elements on the list. In other words `qsort :: (a -> a-> Bool) -> [a] -> [a]` and `qsort (<=)` should work just as the version presented in the book.

4. Write the `merge` and `msort` functions described in problems 7 and 8 on page 72 of the text, except your `msort` function will be a curried function that takes a boolean function just as the `qsort` function in Problem 3.

5. Let us consider an implementation of sets as lists, where each element of a set appears exactly once in a list and the elements appear in no particular order. Do not assume you can sort the lists. Do assume the input lists have no duplicate elements, and do guarantee the output lists have no duplicate elements. Using this implementation, write a curried function to test whether an element is a `member` of a set.

6. Write a recursive curried function to construct the `union` of two sets.

7. Write a recurrisve curried function to construct the `intersection` of two sets.

Extra Credit. Write an efficient function to calculate the nth Pell number:

$$P_n = \begin{cases} 0, & \text{if } n = 0, \\ 1, & \text{if } n = 1, \\ 2P_{n-1} + P_{n-2}, & \text{if } n > 1. \end{cases}$$

Your function must be able to compute the 100,000th Pell number in less than a minute without storing values or returing a list or pair.