Homework 3: Due Wednesday, February 8

Problem 1: Write the following ML functions using pattern matching and recursion.
a. inPosition: Takes two inputs, an α list xs and a nonnegative integer k, and produces the k^{th} element of the list, assuming that we start indexing the list with 0. For example, $\mathsf{inPosition}([5,6,2,4],0) = 5$ and $\mathsf{inPosition}([5,6,2,4],1) = 6$. If k is not less than the length of the list, your program should raise an exception (see Section 1.13).
b. take: Takes two inputs, an α list xs and a nonnegative integer k, and produces the α list consisting of the first k values of xs. If k is the greater than the length of xs, then the function should produce xs. For example, $\mathsf{take}([6,1,3,4],2) = [6,1]$.
c. flatten: Takes one input, an α list list xs (i.e. xs is a list, each of whose elements is an α list), and produces the α list that results from concatenating the elements of xs. For example, $\mathsf{flatten}([[1,2],[3,4,5],\mathsf{nil},[6]]) = [1,2,3,4,5,6]$.
d. zip: Takes two inputs, an α list xs and a β list ys, and produces the $(\alpha * \beta)$ list obtained by pairing the corresponding elements of each list. For example, $\mathsf{zip}([1,2,3],[4,5,6]) = [(1,4),(2,5),(3,6)]$. If one list is longer than the other, then the extra elements should be ignored, i.e. $\mathsf{zip}([1,2],[3,4,5,6]) = [(1,3),(2,4)]$.

Problem 2: Working with sets coded as unordered lists without repetition, write ML functions using pattern matching and recursion that perform the following.
a. intervalSet: Takes two integer inputs m and n, and produces the integer set $\{\mathsf{m}, \mathsf{m}+1, \mathsf{m}+2, \ldots, \mathsf{n}\}$. If $\mathsf{m} > \mathsf{n}$, then your program should produces the empty set.
b. setIntersection: Takes two inputs, assumed to be α sets, and produces the intersection of the inputs.
c. setMax: Takes one input, assumed to be a nonempty set of integers, and produces the maximum of that set. Think about what the appropriate response should be if the list is empty, and act accordingly.
d. setMap: Takes two inputs, a function g of type $\alpha \to \beta$ and an α set xs, and produces the β set obtained by applying g to each element of xs. Ensure that the output is a set!

Problem 3:
a. If $n, d \in \mathbb{N}^+$, we say that d is a *divisor* of n if d divides evenly (i.e. without remainder) into n. Write a program divisorSet that takes one input, assumed to be a positive integer, and produces the set of positive divisors of that number. You should us setFilter along with other functions from Problem 2 and class.
b. Use your work above to write a program gcd that takes two inputs, assumed to be positive integer, and returns the greatest common divisor of the two inputs. In other words, your program should output the largest integer that divides evenly (i.e. without remainder) into both of the inputs.
c. Run your gcd program on inputs of various sizes, and determines where the program starts to take a while. Where do you think the bottleneck is? Why?