CMPSC 461 **Programming Language Concepts** *Spring 2014*
Penn State University **Assignment #2** *Due: 30 January*

The purpose of this assignment is to gain some experience working with lists and programming recursive functions in Scheme. You may not use any of Scheme's imperative features (assignment/loops). You must submit your solution as a source code attachment to the Angel Drop Box.

1. (10 pts) Define two functions `take-first` and `skip-first` that each take a natural number `n` and a list `xs`. The function `take-first` returns the first `n` elements of the list `xs` or the entire list if the its length is less than `n`. The function `skip-first` returns the list obtained from `xs` by "skipping over" or removing the first `n` elements of `xs`, returning the empty list if `n` is greater than the length of the list.

2. (10 pts) Define a function `palindrome?` that takes a list and returns true exactly when the list is a palindrome, i.e., the list of elements reads the same forwards and backwards.

3. (10 pts) Define a single recursive function `distinct` that takes two lists of numbers and returns true exactly when the lists have no numbers in common. You may use built-in list functions, but otherwise your recursion should only involve examining the lists without doing anything else to them.

4. (10 pts) Define a recursive function `better-distinct`, which is a better version of `distinct`, using the following technique. The two lists are first sorted and then the two sorted lists are compared. For this second phase your function must exploit the fact that the lists are sorted. You can use any of the sorting functions provided in class.

5. (10 pts) Define a function `powerset` that takes a list representing a set (you can assume all the elements are unique) and returns the list of all the subsets (each represented as a list). The order of the subsets is not important. Remember that the empty set is a subset of any set. Think carefully about this problem before trying to solve it. Consider the definition of power set and how to construct an inductive (recursive) definition of it from a list representing a set.