Programming Assignment #1 – Due in class on Wednesday, 15 April 2009

Problem Statement:
For this programming assignment, you will gain some initial experience programming in Haskell using modules, lists, simple data types, and polymorphic functions.

Simple Finite Ordered Sets
Define a Haskell module \texttt{FiniteOrdSet} and a polymorphic data type \texttt{Set a} that has a constructor that constructs a type \texttt{Set} from a list of type \texttt{a}. The type \texttt{a} should be both an Equality type (\texttt{Eq}) and an Ordered (\texttt{Ord}) type. You will also define a group of functions, given below, for doing basic operations on sets of elements represented as lists. For example: \([1,2,3,4,5,6,7,8,9,10]\\), \(['a','b','c']\\), [“a”, “list”, “of”, “words”], and [] (the empty list). Note that a set can contain other sets. For example, the \texttt{powerset} of the set \([1,2,3]\) is the set \([[],[1],[2],[3],[1,2],[1,3],[2,3],[1,2,3]\)]. The set operations must include, but are not limited to, the following:

\begin{itemize}
    \item \texttt{set} – given a list of elements all of the same type, the set function removes duplicate elements and orders the elements according to the \texttt{order relation} for elements of that type. For example, the expression set \([2,2,1,3,0,9,1,3]\) returns Set \([0,1,2,3,9]\) as a result. Note that type names start with a capital letter and function names start with a lowercase letter. Ordering the elements improves the efficiency of some of the following operations.
    \item \texttt{empty} – given a set, compute a predicate that decides whether a set is empty.
    \item \texttt{card} – given a set \(S\), compute the cardinality (i.e., size) of the set.
    \item \texttt{member} – given an element \(x\) (which may be a set) and a set \(S\) (which may be a set of sets), compute the predicate that decides whether or not \(x\) is a member of \(S\).
    \item \texttt{equals} – given two sets, compute a predicate that decides whether the two sets are equal. Two sets are equal if they have the same elements.
    \item \texttt{union} – given two sets of the same type, compute the ordered union of the sets.
    \item \texttt{inters} – given two sets, compute the intersection of the sets.
    \item \texttt{product} – given two sets, compute the cartesian product of the members of the two sets.
    \item \texttt{powerset} – given a set \(S\), compute the power set of \(S\).
    \item \texttt{subset} – given two sets \(S_1\) and \(S_2\), compute a predicate the decides whether \(S_1\) is a subset of \(S_2\).
\end{itemize}

For the functions: equals, member, union, inters, product and subset, define corresponding \texttt{left associative infix} operators using the Haskell \texttt{infixl} directive. The infix operators may use different names than the prefix functions; for example: \(3 \ `\texttt{memb}\` [1,2,3] \Rightarrow \texttt{True}\) is the same as member \(3 [1,2,3] \Rightarrow \texttt{True}\).

Submission Requirements:
You are to hand in at the start of class on the due date a complete source code listing of all source files with your name and email address in a comment line at the top of each file. Please use \texttt{enscript -2Gr -C -Ehaskell FinOrdSet.hs} to print your source listing for submission. You are also to submit a gzipped electronic copy of the source file(s) for the program to the course TA via email. Please use the filename: cs209-program-01-firstname.lastname.

Honor Policy:
You may discuss with friends or classmates the general nature of the conceptual problem to be solved, but you are to complete the actual programming for this assignment without resorting to help from any other person or other resources that are not authorized as part of this course. If in doubt, ask the course instructor. \textbf{You may not use the Internet to search for solutions to the problem.}