Due September 26, 1:30 p.m.

Instructions

- Collaboration with your fellow students is not allowed; if you get stuck or have questions, ask me or one of the coaches. (Office hours are posted on the course web site.)
- Submit problems 1–3 on paper.
- For problems 4, remember to document your code and include test cases with the tester function, as you’ve done in labs.
- Submit problem 4 electronically: submit145 asmt2 <your dir>

Problem 1

Recall the definitions of reflexive, symmetric and transitive relations. Let $A = \{1, 2, 3\}$.

a. Give an example of a binary relation $R_1$ over $A$ that is reflexive and symmetric, but not transitive.

b. Give an example of a binary relation $R_2$ over $A$ that is reflexive and transitive, but not symmetric.

c. Give an example of a binary relation $R_3$ over $A$ that is symmetric and transitive, but not reflexive.

Note: A relation is a set of ordered pairs, so your examples should be in that form.

Problem 2

Prove that the intersection of two transitive relations must be transitive.

Problem 3

Give an example of two relations, $R$ and $S$, over the set $A = \{1, 2, 3\}$ such that $R$ and $S$ are transitive, but $R \cup S$ is not transitive. Note: Again, give your answer as a set of ordered pairs.

Problem 4

Define a function, called power-set, that takes a list representing a set as its only input. It should return a list-of-lists representing the power-set of the input list, as illustrated below.

```scheme
> (power-set '(a b c))
((()) (c) (b) (b c) (a) (a c) (a b) (a b c))
```
The order of the subsets doesn’t matter. The built-in \texttt{length} function can be used to help with testing:

\begin{verbatim}
> (length (power-set '(1 2 3 4 5 6 7 8 9 10)))
1024
\end{verbatim}

How to approach the power-set computation:

\textit{Example:}

• listy is \( (1 \ 2 \ 3) \)
• first element is 1
• rest of listy is \( (2 \ 3) \)

A recursive function call generates the power-set of the rest of the list, namely:

\( ((), \ (2), \ (3), \ (2 \ 3)) \)

Then we somehow create another list that is pretty much like this one except that each sub-list has a 1 \texttt{consed} onto it:

\( ((), \ \ (1), \ \ (2), \ \ (3), \ \ (1 \ 2), \ \ (1 \ 3), \ \ (1 \ 2 \ 3)) \)

Then we concatenate these two lists together to yield the desired result:

\( ((), \ (2), \ (3), \ (2 \ 3), \ (1), \ (1 \ 2), \ (1 \ 3), \ (1 \ 2 \ 3)) \)

\textit{Hints:}

• Use a \texttt{let*} to generate a sequence of helpful preliminary results.
• Carefully consider \texttt{cons}, \texttt{list}, and \texttt{append}. Which do you need for what?
• Look at the solutions to Lab 1 and the definition of \texttt{cross-product} from class.