Note: Answers to the exercises listed below should be handed to the instructor in hardcopy and in typewritten form (latex formatting preferred) by the deadline mentioned above. You should also email the Maude code for Problem 2 to fanyang6@illinois.edu.

1. Note that we can think of a relation $R \subseteq A \times B$ as a “nondeterministic function from $A$ to $B$.” That is, given an element $a \in A$, we can think of the result of applying $R$ to $a$, let us denote it $R\{a\}$, as the set of all $b$’s such that $(a, b) \in R$. Unlike for functions, the set $R\{a\}$ may be empty, or may have more than one element.

Note that the powerset $\mathcal{P}(B)$ allows us to view the “non-deterministic mapping” $a \mapsto R\{a\}$ as a function from $A$ to $\mathcal{P}(B)$. More precisely, we can define $R\{\_\}$ as the function:

$$R\{\_\} : A \ni a \mapsto \{ b \in B \mid (a, b) \in R \} \in \mathcal{P}(B).$$

But since this can be done for any relation $R \subseteq A \times B$, the mapping $R \mapsto R\{\_\}$ is then a function:

$$\_\{\_\} : \mathcal{P}(A \times B) \ni R \mapsto R\{\_\} \in [A \rightarrow \mathcal{P}(B)].$$

One can now ask an obvious question: are the notions of a relation $R \in \mathcal{P}(A \times B)$ and of a function $f \in [A \rightarrow \mathcal{P}(B)]$ essentially the same? That is, can we go back and forth between these two supposedly equivalent representations of a relation? But note that the idea of “going back and forth” between two equivalent representations is precisely the idea of a bijection.

Prove that the function $R\{\_\} : \mathcal{P}(A \times B) \ni R \mapsto R\{\_\} \in [A \rightarrow \mathcal{P}(B)]$ is bijective.

2. This problem is a good example of the motto:

\[\text{Declarative Programming = Mathematical Modeling}\]

Specifically, of how you can model discrete mathematics in a computable way by functional programs in Maude, so that what you get is a computable mathematical model of discrete mathematics. Furthermore, it will allow you to obtain a computable mathematical model of arrays and array lookup as a special case of your model.

Recall the function $R\{\_\} : \mathcal{P}(A \times B) \ni R \mapsto R\{\_\} \in [A \rightarrow \mathcal{P}(B)]$ from Problem 1 above. Note that we then also have a function:

$$\_\{\_\} : \mathcal{P}(A \times B) \times A \ni (R, a) \mapsto R\{a\} \in \mathcal{P}(B)$$

that applies the function $R\{\_\}$ to an element $a \in A$ to get its image set under $R$.

Define this latter function in Maude for $A = \mathbb{N}$ the set of natural numbers, and $B = \mathbb{Q}$ the set of rational numbers, and for finite relations $R \subset \mathbb{N} \times \mathbb{Q}$ by giving recursive equations for it in the functional module below.

Define also in the same functional module the auxiliary functions: dom, which assigns to each finite relation $R \subset \mathbb{N} \times \mathbb{Q}$ the set $\text{dom}(R) = \{ n \in \mathbb{N} \mid \exists (n, r) \in R \}$, and the predicate $\text{pfun}$, which tests whether a relation $f \subset \mathbb{N} \times \mathbb{Q}$ is a partial function. That is, whether $f$ satisfies the uniqueness condition:

$$(\forall n \in \mathbb{N}) \ (\forall p, q \in \mathbb{R}) \ [((n, p) \in f \land (n, q) \in f) \Rightarrow p = q].$$

Note that the function $R\{\_\}$ is closely related to the function

$$R\{\_\} : \mathcal{P}(A) \ni A' \mapsto \{ b \in B \mid a \in A' \land (a, b) \in R \} \in \mathcal{P}(B)$$

defined in STACS, namely, by the equation: $R\{a\} = R\{\{a\}\}$. We are using a different notation ($R\{\_\}$ and $R[\_]$) to distinguish them.
In Computer Science a \textit{finite} partial function $f \subset \mathbb{N} \times \mathbb{Q}$ is called an \textit{array} of rational numbers, or sometimes a \textit{map}. Note that when $f$ is an array, the result $f\{n\}$ is either a single rational number, or, if $f$ is not defined for the index $n$, then $\text{mt}$. That is, $f\{n\}$ is \textit{exactly} array lookup, which usually would be denoted $f[n]$ instead. In summary, the function $\_\{\_\}$ that you will define includes as a special case the \textit{array lookup} function for arrays of rational numbers of arbitrary size.

\textbf{Note}: Notice Maude's built-in module \texttt{RAT} contains \texttt{NAT} as a submodule, and has a subsort relation \texttt{Nat < Rat}. You can use the automatically imported module \texttt{BOOL} and its built-in equality predicate $==$ and if-then-else \texttt{if\_then\_else\_fi} as auxiliary functions.

\begin{verbatim}
  fmod RELATION-APPLICATION is protecting RAT .
  sorts Pair NatSet RatSet Rel .
  subsort Pair < Rel .
  subsort Nat < NatSet < RatSet .
  subsort Rat < RatSet .
  op [_&_] : Nat Rat -> Pair [ctor] . *** Pair is cartesian product Nat x Rat
  op mt : -> NatSet [ctor] . *** empty set of naturals
  op null : -> Rel [ctor] . *** empty relation
  op _&_ : NatSet NatSet -> NatSet [ctor assoc comm id: mt] . *** union
  op _&_ : RatSet RatSet -> RatSet [ctor assoc comm id: mt] . *** union
  op _&_ : Rel Rel -> Rel [ctor assoc comm id: null] . *** union
  op _in_ : Nat NatSet -> Bool . *** membership
  op _{\_} : Rel Nat -> RatSet . *** relation application to a number
  op dom : Rel -> NatSet . *** domain of a relation
  op pfun : Rel -> Bool . *** partial function predicate
  vars n m : Nat . var r : Rat . var P : Pair . var S : NatSet . var R : Rel .
  eq n,n = n . *** idempotency
  eq P,P = P . *** idempotency
  eq n in mt = false . *** membership

  eq n in (m,S) = (n == m) or n in S . *** membership

  *** your equations defining the functions _{\_}, dom, and pfun here
  *** if you need to declare any other variables or auxiliary
  *** functions besides those above, you can also do so

endfm

You can retrieve this module as a “skeleton” on which to give your answer from the course web page. Also, send a file with your module to fanyang6@illinois.edu.
\end{verbatim}