In-Class Problems Week 2, Fri.

Problem 1. Subset take-away[1] is a two player game involving a fixed finite set, A. Players alternately choose nonempty subsets of A with the conditions that a player may not choose

- the whole set A, or
- any set containing a set that was named earlier.

The first player who is unable to move loses the game.

For example, if A is $\{1\}$, then there are no legal moves and the second player wins. If A is $\{1, 2\}$, then the only legal moves are $\{1\}$ and $\{2\}$. Each is a good reply to the other, and so once again the second player wins.

The first interesting case is when A has three elements. This time, if the first player picks a subset with one element, the second player picks the subset with the other two elements. If the first player picks a subset with two elements, the second player picks the subset whose sole member is the third element. Both cases produce positions equivalent to the starting position when A has two elements, and thus leads to a win for the second player.

Verify that when A has four elements, the second player still has a winning strategy.[2]

[1]From Christenson & Tilford, *David Gale's Subset Takeaway Game, American Mathematical Monthly, Oct. 1997*

[2]David Gale worked out some of the properties of this game and conjectured that the second player wins the game for any set A. This remains an open problem.

Problem 2. **(a)** Define a bijection between the positive integers and all integers.

(b) Define a bijection between the positive integers and $\mathbb{Z} \times \mathbb{Z}$, the set of all the ordered pairs of integers:

$$
\begin{array}{l}
(0,0),(0,1),(0,-1),(0,2),(0,-2),(0,3),(0,-3),\ldots \\
(1,0),(1,1),(1,-1),(1,2),(1,-2),(1,3),(1,-3),\ldots \\
(-1,0),(-1,1),(-1,-1),(-1,2),(-1,-2),(-1,3),(-1,-3),\ldots, \\
(2,0),(2,1),(2,-1),(2,2),(2,-2),(2,3),(2,-3),\ldots \\
(-2,0),(-2,1),(-2,-1),(-2,2),(-2,-2),(-2,3),(-2,-3),\ldots \\
\vdots
\end{array}
$$

(c) Conclude that the set of positive integers is the same size as the set of all rational numbers.

Problem 3. Consider procedures in your favorite programming language — say, C++, Java, or Scheme — that can take a string of ASCII characters as an argument. Call a procedure of this kind a *string procedure*.

A set, $\mathcal{R}$, of ASCII strings is called *recognizable* (using your programming language) if there is a string procedure that returns the integer 1 when it is applied to a string that is in $\mathcal{R}$, and does not return 1 (it need not return any value at all) when it is applied to a string that is not in $\mathcal{R}$.

Now any program can be represented by a string of ASCII characters. So if, in particular, a string s is the definition of a string procedure, let's refer to that procedure as P_s. In case string s is *not* a definition of a string procedure, let's define P_s to be some fixed string procedure that never returns a value no matter what string it's applied to. In this way, every string, s, gets associated with a (frequently useless) string procedure, P_s.

We're going to define a set, $\mathcal{V}$, of strings in a way similar to the set in Russell's Paradox. Namely, let $\mathcal{V}$ be the set of strings, s, such that P_s applied to argument s does *not* return the integer 1. So for every ASCII string, s, we have by definition

$$s \in \mathcal{V} \quad \text{iff} \quad P_s \text{ applied to } s \text{ does not return 1.} \tag{1}$$

Prove that $\mathcal{V}$ is not recognizable.