INTRODUCTION TO TYPE SYSTEMS.

Problem 1. We often want to perform operations on objects of different types. For instance, we might want to add an integer and a rational number.

For the present conversation, we will restrict our domain to numeric types.

Please create a diagram that organizes the following numeric types in some sort of hierarchical and/or enclosed/intersecting sets representation.

- integers
- rational numbers
- a complex number represented in rectangular form (i.e., $x + yi$)
- a complex number represented in polar form (i.e., $r$ and $\theta$)
- a complex number, regardless of underlying representation
Problem 2. Make a list of some numeric conversion operations you may need to implement to support the numeric types listed previously.

Do you have an exhaustive list?

Can you shorten your list and still be able to convert among types as needed?

Problem 3. What are some issues that might arise in type conversions?
**Problem 4.** In usual compiled languages (C/C++; Java), objects are declared with a specific type. At compile time, the compiler will select pre-existing routines to convert among types, or exit with an error if there isn’t a way of doing conversions. Successful conversions may be selected either automatically or with the use of programmer-specified type coercions.

On the other hand, in a dynamic programming language like Scheme, objects need to be combined at runtime.

List some approaches that may be necessary to support this.