Problem 1.2.1. Pick your favorite programming language. Is it interpreted? Is it compiled? What is (or are) the data type(s) for floating point numbers? Is there a built-in data type for vectors or matrices? Can the vectors (or matrices) be added? Can they be multiplied?

Problem 1.2.2. What facilities are included in your favorite programming language for parallel programming? Are there libraries that provide these facilities? Is it possible to perform both shared memory and distributed memory computing in your language? With these libraries?

Problem 1.2.3. Is your favorite programming language garbage collected? That is, is there automatic de-allocation of unusable objects? What are the advantages and disadvantages of garbage collection?

Problem 1.2.4. The following function applies the function f to each item of a vector of items. Implement it in your favorite programming language.

```
function MAP(f, x)
    y ← new array the same size as x
    for i an index of x do
        y_i ← f(x_i)
    end for
end function
```

Now implement it in your least favorite, or a previously unknown, programming language.

Problem 1.2.5. Macros are pieces of code that *transform other pieces of code* before compilation or execution. Does your favorite programming language have macros? What kinds of transformations can themacros in that language perform? If your favorite programming language does not have macros, or you discover another language that does have macros (such as Lisp or Julia), describe the macro facilities of that language.

Problem 1.2.6. The C programming language was developed for implementing the Unix operating system. This led to many design decisions by the creators of the language, such as pointer arithmetic, and lack of array bounds. Explain why these decisions may have been necessary for their purpose at that time, and if you think those decisions help or hinder the creation of mathematical software now.

Problem 1.2.7. Some languages are stack languages, such as Forth, Joy, and PostScript. Arguments for a "function" in a stack language do not have names, but the $k-th$ input is the $k-th$ item from the top of the stack. Download one of these languages and implement the "map" function of Problem 1.2.4 in that language.