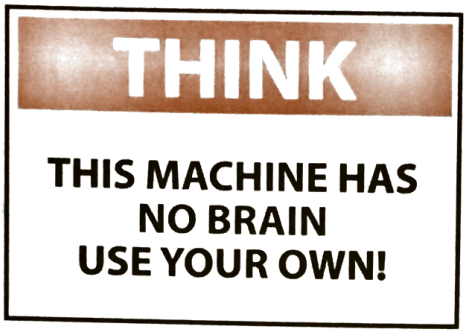
joins the points with a line (whose style can be set with `PlotStyle`), specifies the range of the horizontal and vertical axes to show on the plot, and adds a label.

A.8 DEBUGGING

Problems are inevitable—your carefully constructed program will usually give you errors the first time you run it. As you attempt to fix it, the most important thing to remember is



Here are some more specific hints and tips to help you navigate the programming process:

1. *Think first, code second.* Be sure you understand the problem before you start writing code. It can be helpful to write (in human-readable prose) a plan outlining what your program will do. Use this plan to write a set of comments, and then fill in the calculation between the comments.
2. *Use meaningful variable names and comments* to explain what you are doing. This helps avoid mistakes, and also makes it easier for somebody else to understand your code.
3. *Keep track of units.* It is generally better to select a unit system that avoids very large or very small numbers in order to prevent arithmetic rounding error, and to use a consistent unit system throughout the calculation. For many properties, SI units are *not* the correct choice, since the numbers are very small.
4. *Don't reinvent the wheel.* Use built-in functions whenever possible. Use the online documentation to confirm you are using these functions correctly.
5. Divide your program into *small, reusable functions*. This makes it easier to test and reuse code in future programs.
6. *Test functions to make sure they work before putting them together into larger assemblies.* Ways to test functions include comparison to exactly known results, comparison to independent implementations by other individuals, and looking at graphs/charts for reasonable behavior.
7. *Build complicated expressions from the inside out.* Large sets of nested expressions can get quite unwieldy. A good strategy is to begin by writing the inside of loops and other calculations first, test the operation with a fixed value, and then add the necessary loop structures around this existing code.
8. *Make sanity checks*, i.e., test cases that you know the answer to because it is simple (and you have worked it out by hand) or because it is in a published source (and someone

else presumably spent a lot of time making sure the answer is correct). Be sure that you can reproduce a few of these before moving on to do new science. This applies to both individual functions and entire programs.

9. *Visualize success.* A picture (of your results) is worth a thousand columns of numbers. This is true both for debugging and for presenting your work to others.
10. *Variable name typos* and *uninitialized variables* (variables whose value has not yet been set) are common sources of error. A subset of this problem is the use of variable or function names that clash with names of built-in constants and variables (e.g., `E`). Mathematica notebooks color-code variables: local variables are green, loop variables are cyan, defined global variables are black, and undefined variables are blue. The rendering used to describe these in print is shown on page xxvii in the Preface.
11. Don't confuse *assignment versus equality*; see Section A.4.2. Check variable assignment and `If []` statements to make sure that the program is doing what you want it to do.
12. *Uncover assumptions the hard way.* One debugging strategy is to sit down with another human and explain what is being done for a particular example case (e.g., particular explicit values of inputs), stepping through each line of the program. Use a piece of scrap paper to store the value of each of the variables, and "update" them by hand. Pay attention to what the program literally says, not what you think it is supposed to be doing. Most problems arise because your assumptions about what the code is doing are not the same as what it is actually doing. (For that matter, this is the origin of many problems in life too.) Often, the act of having to explain your code will cause you to have a new insight into why it does not work.
13. *Start early and take your time.* It is hard to find bugs when under deadline pressure. Sometimes going for a walk, going to bed, etc., and approaching the problem with a fresh set of eyes (and fresh brain) is the best way to solve the problem.

PROBLEMS

A-1. Write logical expressions to express the following conditions:

- **(a)** `X` is greater than 4
- **(b)** `X` is between 3 and 10
- **(c)** `X` is less than 3 or greater than 10
- **(d)** `X` and `Y` are both positive
- **(e)** `X` and `Y` are equal to each other
- **(f)** `X` is less than 2 or `Y` is less than 2, but not both

A-2. Define a function to calculate the energy of an electron in a hydrogenic atom that takes arguments of atomic charge, Z, and principal quantum number n^2. (Recall: In atomic units, $E(Z, n) = (-1/2)(Z^2/n^2)$). Demonstrate that your function works by calculating the energies when $Z = 1$ and $n = 1, 2, 3, 4$.