In the first lab, we ask the computer to generate sequences of numbers

$$x_0,\ x_1,\ x_2,\ \ldots$$

where $x_1 = f(x_0)$, $x_2 = f(x_1)$, and in general $x_{i+1} = f(x_i)$ for a function f given by $f(x) = ax + b$. We'll need a program that takes as input the parameters a and b, along with a starting value x_0, and, since we won't have time to look at an entire infinite sequence, a number n of terms to generate.

Here are three fairly easy ways to do this.

Option I: Run the TrueBASIC program ITERLIN (Math Courses ▷ Math 251 ▷ CHAP1)

The program listed on p. 14 of the textbook is available on the lab computers. In the CHAP1 folder, double-click "ITERLIN" (the one with two tiny graphs), and you'll get a window showing you the BASIC code and offering you the opportunity to run the program.

Option II: Write a Maple procedure (Programs ▷ Math ▷ Maple 9)

You've probably used Maple as a sort of super-calculator that evaluates whatever expressions you type into it. You can also write short programs, called "procedures," in the Maple environment. Thus, you can use Maple's mathematical muscle in a repetitive way without doing huge amounts of typing.

The `iterlin()` procedure (code to follow) doesn't use much heavy mathematics, but it demonstrates how to get Maple to do something repetitive for you.

```
iterlin := proc(a, b, x0, n)
local i,x;
x := x0:
for i from 1 to n do
x := a*x + b:
print(x):
end do:
end proc:
```

As you type in the procedure, you should end each line with SHIFT-ENTER. If you don't, Maple will try to interpret the procedure before you've finished typing it. Type just ENTER only at the end of the last line. Maple will then read in the procedure.

For the rest of this Maple session, when you type `iterlin(`a`, `b`, `x_0`, `n`);`, Maple will execute the procedure with the given values. For example, if you enter

```
iterlin(0.5, 2, 5, 10);
```

Maple will give a list of the first ten terms of the sequence

$$x_0, x_1 = f(x_0), x_2 = f(x_1), x_3 = f(x_2), \ldots$$

with $x_0 = 5$ and $f(x) = 0.5x + 2$.

Option III: Write a *Mathematica* function (Programs ▷ Math ▷ Mathematica 5)

Mathematica is a little less friendly to use than Maple, but it is somewhat more powerful, and worth learning. Moreover, *Mathematica* has a built-in function iteration routine (called `NestList[]`) which makes writing a *Mathematica* version of `iterlin()` almost trivial.

To define the function `iterlin` in *Mathematica*, type in

```
iterlin[a_, b_, x0_, n_] := NestList[(a #1 + b)&, x0, n]          (1)
```

followed by SHIFT-ENTER.

(In *Mathematica*, you type SHIFT-ENTER at the end of an expression to tell the program to interpret it. If you want to start a new line in the middle of your input, you type just ENTER, and the program waits for more. The convention in Maple is exactly the opposite: SHIFT-ENTER means there's more input coming, and ENTER means that the input is complete, and the program should try to interpret it. Another difference between the programs: Maple uses parentheses for function arguments; *Mathematica* uses square brackets.)

To generate the terms in the sequence from the example above, type

```
iterlin[0.5, 2, 5, 10]
```

followed by SHIFT-ENTER.

Option IV: Use an Excel spreadsheet (Programs ▷ Microsoft Office ▷ Excel)

Spreadsheet programs are built for repetitive calculations, and if the individual calculations are fairly simple (as in this case), you might want to use Excel to do your experiments.

Here's one way to set up a spreadsheet that does the `interlin` calculation: Type "a=" in cell A1, "b=" in cell A2, and "x0=" in cell A3. You'll put your values for a, b, and x_0 into cells B1, B2, and B3, respectively. In fact, do that now. To repeat the example above, type 0.5 in cell B1, 2 in cell B2, and 5 in cell B3.

We'll use column C for the calculations. To put the value of x_0 in cell C1, type "=$B$3" in cell C1. We want C2 to contain

$$a \times (\text{value in C1}) + b$$

so we type "=$B$1*C1+$B$2" into cell C3. (The cell references $B$1 and $B$2 are absolute, meaning "use the values in the actual cells B1 and B2"; the reference C1 is relative, meaning "use the value in the cell just above this one.") Now we want to copy this formula into a column of cells below C2. To do this, highlight a column from C2 downward (to, say, C30), and press CTRL-D.

Now any time you change the value of a, b, or x_0, Excel will quickly recalculate the first 30 terms of the sequence

$$x_0, x_1 = f(x_0), x_2 = f(x_1), x_3 = f(x_2), \ldots$$

Appendix: That weird stuff in Option III

The syntax of our definition of `interlin[]` in *Mathematica* deserves some comment. First, you need to know the syntax of the built-in function `NestList`. It takes three arguments: `NestList[f, x0, n]`, and returns a list of the first `n` terms in the sequence

`x0`, `f[x0]`, `f[f[x0]]`, `f[f[f[x0]]]`, ...

(Type `?NestList` SHIFT-ENTER in *Mathematica* to see where this comes from.)

We define `iterlin` to be a function with four arguments called `a`, `b`, `x0`, and `n`. The underscore after each of these letters on the left side of (1) tells *Mathematica* that it's the name of an argument to a function. Note that on the right side, we don't type the underscores. This is because we want the right side to use the values (rather than the names) of the arguments.

The second and third arguments to `NestList` should be clear: we just tell `NestList` to use the starting value and the number of terms that were specified to `iterlin`. The first argument is the tricky one. `NestList` expects a function in this position, and the expression `(a #1 + b)&` is a *Mathematica* function. The `&` sign says "whatever comes before this is a function," and the `#1` means "the first argument." The `a` and `b` are the values we get from the call to `iterlin`. So the whole expression `(a #1 + b)&` is a function that takes one argument (represented by `#1`), multiplies it by the value of `a`, adds `b`, and returns the result. That is, it's the function f given by $f(x) = ax + b$.