Introduction to Programming with Mathematica, 3rd Edition

First printing
Errata last updated: 2 April 2006

1 Introduction

- **Page 11**, first text line starting with "Here is a short program…", replace this text with:

  "Here is a short program that creates a button containing the above two expressions. (We will explain the need to wrap `Symbol["…"]` around the expressions here in Chapter 10.)"

  Then replace following code with:

```
ButtonBox["Spikey",
   ButtonFunction:→CompoundExpression[
    Needs["Graphics`Polyhedra`"],

Show[Symbol["Stellate"][Symbol["Polyhedron"][Symbol["Icosahedron"]
]]]],
ButtonEvaluator→Automatic, Active→True] //DisplayForm
```

Delete paragraph beginning "The formatted version of the…".

Change paragraph starting "Clicking the button…" to: "Clicking the button will cause the *Mathematica* code in the `ButtonFunction` to be evaluated and the graphic of the icosahedron will then be displayed in your notebook."

3 Lists

- **Page 55**, 9 text lines from top: change "…and the *The Mathematica Book*…" to "…and *The Mathematica Book*…".

4 Functional programming

- **Page 95**, Exercise 4: change `interleave2` to `interleave`.

7 *Recursion*

- **Page 198**, Input In[3]: change to the following (switching `b` and `ra` in the third line of code:

```
In[3]:= merge[{a_, ra___}, {b_, rb___}] :=
         If[a ≤ b,
          Join[{a}, merge[{ra}, {b, rb}]],
          Join[{b}, merge[{a, ra}, {rb}]]
         ]
```

  All of the remaining inputs that use `merge` and `MergeSort` will now be correct.

- **Page 205**, In the text paragraph immediately above Figure 7.1, change the parenthetical "(B)" to "(b)".

9 Graphics programming

- **Page 278**, Exercise 5: change "…appeared in Porta, Davis and Uhl, 1994." to "…appeared in the Calculus&*Mathematica* courseware (Porta, Davis and Uhl 1994)."
.

- **Page 298**, Exercise 13: change "…from Exercise 13…" to "… from Exercise 12…".

10 Front end programming

- **Page 336**, text paragraph beginning "You can use any *Mathematica*…": change `CompoundExpression[Get["Graphics`", LogPlot[Exp[x],{x,1,2}]]` to `CompoundExpression[Get["Graphics`"], LogPlot[Exp[x],{x,1,2}]]`. That is, add closing bracket to `Get["Graphics`"`.

- **Page 337**, Replace the contents of this page starting with the text “Here is the button code…” with the following:

Here is the button code. Note that we have added an option to `ButtonBox` to set the background and also wrapped the `ButtonBox` text in `StyleBox` in order to add several font options there.

```
ButtonBox[StyleBox["Spikey",
      FontFamily→"Times",FontColor→White],
    ButtonFunction:→CompoundExpression[
        Needs["Graphics`Polyhedra`"],

Show[Graphics3D[Symbol["Stellate"][Symbol["Icosahedron"][]]]]],
    ButtonEvaluator→Automatic,
    Background→GrayLevel[.5],
    Active→True] //DisplayForm
```

Spikey

Clicking the button will produce the stellated icosahedron as displayed above.

Note that we have wrapped each of the symbols `Stellate` and `Icosahedron` in `Symbol["…"]`. The reason this is necessary is a bit technical but is roughly as follows: Prior to evaluating any input, the kernel first parses that input and determines which context any symbols in that input live. So prior to evaluation, the symbols `Stellate` and `Icosahedron` are parsed and put in the current context (typically `Global``). Then, when the evaluation takes place and the package is loaded, shadowing of symbols occurs and a warning message will be issued. By using `Symbol["Stellate"]`, the string argument prevents the parser from putting a symbol in the wrong context. Actually, this code would be more robust if we used the full context name, `Symbol["Graphics`Polyhedra`Stellate"]`, as this would work even if shadowing had occurred elsewhere.

11 Examples and applications

- **Page 350**, Exercise 3: change “In addition, you function…” to “In addition, your function…”.

12 Writing packages

- **Page 413**, first line: change “A 100 step off-lattice…” to “A 1000 step off-lattice…”.

- **Page 415**, first line: change "…a package `CollatzSequence.m`…" to "…a package `Collatz.m`…".

References

- **Page 428**: add Lagarias, J., Miller, V., and Odlyzko, A. Computing $\pi(x)$: The Meissel-Lehmer Method. *Math Comp.*, **44** 537–560, 1985.

- **Page 429**: add Porta, H., Davis, W., and Uhl, J. *Calculus&Mathematica*. Reading, MA: Addison-Wesley, 1994.

Solutions to exercises

- **Page 441-2**, Solution 1: change `pairSum` to `addPair` (two occurrences).

- **Page 445**, Solution 3: change "…`Times` functions…" to "…`Times` function…".

- **Page 446**, Solution 4: change `interLeave2` to `interleave`.

- **Page 463**, Solution 9: the `bisect` function, as given in the text, is only valid for functions that are monotonically increasing. The following code corrects for the more general cases.

```
In[31]:= bisect[f_, {a_, b_, ϵ_}] := Module[
                       a + b
           {midpt = N[ ----- ], low = a, high = b}, While[Abs[f[midpt]] > ϵ,
                         2
            If[Sign[f[low]] == Sign[f[midpt]], low = midpt, high = midpt];
                        low + high
            midpt = N[ ---------- ]];
                            2
           midpt]

In[32]:= f[x_] := Cos[x] - x

In[33]:= bisect[f, {0, 1, 10^-12}]

Out[33]= 0.739085

In[34]:= f[x_] := x^2 - 2

In[35]:= bisect[f, {1, 2, 10^-12}]

Out[35]= 1.41421
```

```
In[36]:= f[x_] := 4 - x^2

In[37]:= bisect[f, {0, 4, 10^-12}]

Out[37]= 2.
```

- **Page 483**, last text sentence: delete “The second form simply calls the first.”

- **Page 508**, first line: add “(*Note*: `Graphics`Polyhedra`` calls `Geometry`Poly‐ topes`` in which some of the functions such as `Cube`, `Dodecahedron`, and others are defined. So you might want to examine `Names` of that package as well.)” immediately following this first line.

- **Page 514**, Solution 9: change “Exercise 9” to “Exercise 8”.

- **Page 523–525**, Solution 1: add the following text to the end of the second text paragraph and then modify the `ShowTable` code as below:

  “We also need to account for the situation when no headings are specified; this is done with the `If` statement inside the `GridBox`.”

```
In[7]:= Options[ShowTable] = {Headings → {}};

In[8]:= ShowTable[data_, opts___?OptionQ] :=
         Module[{headstyle, headings},
          headstyle[str_] := StyleBox[(MakeBoxes[#, StandardForm] &)@
              str, FontFamily → "Helvetica",
            FontWeight → "Bold", FontColor → Blue, FontSize → 10];
          headings = Headings /. Flatten[{opts}] /. Options[ShowTable];
          DisplayForm[StyleBox[
            GridBox[If[headings == {}, data,
              Prepend[data, Map[headstyle, headings]]],
             GridFrame → 2, GridFrameMargins → {{1, 1}, {1, 1}},
             RowLines → 1, ColumnLines → 1],
            FontFamily → "Times",
            Background → GrayLevel[.8], SingleLetterItalics → True]]]
```

Index

- **Page 546** add index entry: Perl language, 162