Errata for

Programming, Problem Solving, and Abstraction with C (revised 2013 edition)

by Alistair Moffat

as at **October 6, 2014**

Chapter 1 Computers and Programs

nothing yet

Chapter 2 Numbers In, Numbers Out

page 22
"is finished when an character that cannot be incorporated" should be "is finished when a character that cannot be incorporated"

page 27
Exercise 2.4: Both `limits.h` and `float.h` are required in order to access the set of six constants that are listed there. And you need to use `%e` format to print `FLT_MIN` and `DBL_MIN`; `%f` will just show them as being zero.

Chapter 3 Selection

nothing yet

Chapter 4 Loops

page 51
The last line, "more important that the particular style", should say "more important than the particular style".

Chapter 5 Functions

nothing yet

Chapter 6 Functions and Pointers

nothing yet

Chapter 7 Arrays

nothing yet

Chapter 8 Structures

page 129
In Figure 8.1, there should be a semicolon after the declaration of `name`.

page 131

"structures cannot be compared for equality, even of they have the same type" should be "structures cannot be compared for equality, even if they have the same type"

Chapter 9 Problem Solving

page 155

The asymmetry in the way the `if` guard is presented means that the `bisection` function does not converge if the situation is reached in which `f(mid)==0`; that is, if `mid` becomes the exact root by luck while the search is proceeding. To fix the problem, the code needs to deal with three distinct cases rather than two:

```
if (fx1*fmid < 0) {
    /* root is to left of middle */
    x2 = mid;
    fx2 = fmid;
} else if (fx1*fmid > 0) {
    /* root is to right of middle */
    x1 = mid;
    fx1 = fmid;
} else {
    x1 = x2 = mid;
}
```

page 160

Exercise 9.3, "Write a program that deals four random five-card poker hand" should be "Write a program that deals four random five-card poker hands"

Chapter 10 Dynamic Structures

nothing yet

Chapter 11 Files

nothing yet

Chapter 12 Algorithms

page 206

In Figure 12.1 the midpoint of the search range is found using `mid = (lo+hi)/2`. If the array `A` has a size that approaches the maximum value that can be stored in an integer (and yes, if your computer has enough memory to store an array of that size!), that arithmetic may overflow. If this risk is a concern, then it is more robust to make use of the equivalent computation `mid = lo+(hi-lo)/2`.

page 225
“As a third choice, heap sort is also available . . . a little slower than both quick sort and heap sort” should be “As a third choice, heap sort is also available . . . a little slower than both quick sort and merge sort”.

Chapter 13 Everything Else

nothing yet