Chapter/Page/Line	Original Text	Change To
Ch 3, p 43, line 16-17	The period of a pendulum's swing depends on ~~the weight of the pendulum and~~ its length	The period of a pendulum's swing depends on its length
Ch 4, p 73, line 2	The controlled statement on ~~line 5~~...	The controlled statement result = result + s[i]; on the fifth line...
Ch 4, p 73, line 22	...compiler to call result's move constructor instead of its copy constructor.	...compiler to call result's move assignment operator.
Ch 4, p 85, line 16	...(iterators, indexing, C-string access, simplicity)...	...(iterators, indexing, C-string access, value semantics, simplicity)...
Ch 6, p 123, line 22:	cyclenode fourth = { "4", &first);	cyclenode fourth = { "4", &first }; *Right parenthesis should be right curly brace*.
Ch 6, p 125, line 13	...that knows which of the two ways the shared pointer was constructed.	...that knows whether the sharred object and reference count were allocated as one object or two.
Ch 7, p 152, line 6-7	char* s = "sample data with spaces"; ...	Remove lines
Ch 7, p 152, line 13-14	char* s = "sample data with spaces"; ...	Remove lines

Ch 7, p 152, line 23-29	char* s = "sample data with spaces";
	size_t i;
	...
	for (i = 0; i < strlen(s); ++i)
	if (s[i] == ' ')
	strcpy(&s[i], &s[i+1]); // remove space
	s[i] = '\0';
		for (size_t i = 0; i <= strlen(s); ++i)
		if (s[i] == ' ')
		strcpy(&s[i], &s[i+1]); // copy string tail over space

Ch 7, p 152, line 33-35	(The repeated call to strlen() is not the only thing ~~less than optimal about~~ this function. Other ~~optimizations~~ are left as an exercise.)	(The repeated call to strlen() is not the only thing wrong with this function. Other improvements are left as an exercise.)
Ch 7, p 153, line 2	In Example 7-8, the function strlen() is a pure function.	In Examples 7-7 and 7-8, the function strlen() is a pure function.
Ch 7, p 153, lines 2-3	In the first loop, its argument	In Example 7-7, its argument s

Chapter/Page/Line	Original Text	Change To
	s is never modified by the loop, so the call to strlen() is loop-invariant. In the second loop, the call to strcpy() modifies s, so the call strlen(s) is not loop-invariant.	is never modified by the loop, so the call to strlen() is loop-invariant, and can be pulled out of the loop. In Example 7-8, the call to strcpy() modifies s, so the call strlen(s) is not loop-invariant.
Ch 7, p 166, line 33	To implement polymorphic behavior,…	To implement polymorphic behavior in a C++ program, a base class defines...
Ch 7, p 166, line 34-36	A base class *defines* the interface by declaring a set of pure virtual functions ~~(functions with declarations, but no function body). Because pure virtual functions have no body,~~ C++ prevents the interface base class from being instantiated. Derived classes *implement* the interface by providing overrides (definitions)...	A base class *defines* the interface by declaring a set of pure virtual functions. C++ prevents the interface base class from being instantiated. Derived classes *implement* the interface by providing overrides (declarations and definitions)...
Ch 7, p 177, line 13	...the integer expression x*4 can be recoded more efficiently as x<<2.	...the integer expression x*8 can be recoded more efficiently as x<<3, because 8 is 2^3, and because shifting left *k* bits is equivalent to multiplying by 2^k.
Ch 7, p 177, line17-20	If one argument or the other can be modified to provide the exponent rather than the power of two value, the developer can...	If the developer can modify the program to provide the exponent *k* rather than 2^k, he can...
Ch 9, p 217, line 12	that matched key. *"key" is in typewriter font*	that matched value. *"value" is in typewriter font*
Ch 9, p 222, line 14-15	~~Of course, with operator size_t() hijacked for this use, it isn't available to return the size of a charbuf. The expression sizeof(charbuf) will return very misleading data.~~	
Ch 9, p 225, line 11-13	~~if (key[0] < 'a' || key[0] > 'z') return 0; return (key[0]-'a');~~	return key[0] % 26;
Ch 9, p 225, line 21-22	~~This is defensive programming to prevent accessing undefined storage in case the key was something like "@#$%".~~	
Ch 10, p 252, line 9	The storage allocated for each ~~list item~~ is...	The storage allocated for each item in the map is...
Ch 10, p 254, line 16	// key ~~not found~~ path	// key inserted path

Chapter/Page/Line	Original Text	Change To
Ch 10, p 254, line 24	...~~found~~ and false if it was ~~inserted~~.	...inserted and false if it was found.
Ch 10, p 254, line 35	...~~lower~~_bound() for a C++98-style hint, or ~~upper~~_bound()...	...upper_bound() for a C++98-style hint, or lower_bound()...
Ch 10, p 254, line 36	...whose key is less...	...whose key is *not less*...
Ch 10, p 256, line 8-9	...the items returned by lookup are const.	...the items returned by lookup are const in an std::set.
Ch 11, p 271, line 18	Example 11-7 is a version of stream_read_string()...	Example 11-7 is a version of stream_read_string_reserve()...
Ch 12, p 300, line 15	std::cout << "total items ~~consumed~~ " << counter << std::endl;	std::cout << "total items produced " << counter << std::endl;
Ch 12, p 318, line 1	The disk drive can only be...	The disk drive's read head can only be...
Ch 13, p 338, line 37-38	~~block_arena is a fixed pool of memory that can be allocated by block_manager.~~	fixed_arena_controller provides a single static block of memory from which fixed-size nodes can be allocated.
Ch 13, p 346, line 36-37	The default constructor is typically empty in stateless ~~constructors~~...	The default constructor is typically empty in stateless allocators...