Draft Technical Corrigendum
ISO/IEC 13816:1996(E)
Information technology–Programming languages, their environments and system software interfaces–Programming Language ISLISP

1. *Page 1, before section 1.1*

   Add chapter number and title "**1 Scope, Conventions and Compliance**". [Note: It was lost during the final publication editing.]

2. *Page 1, section 1.2*

   Replace the second reference "IEEE standard 754-1985. *IEEE standard for Binary floating point arithmetic. IEEE, New York, 1985*" with "IEEE standard 754-1985. *Standard for binary floating point arithmetic*". [Note: Use the standard format.]

3. *Page 2, 3rd paragraph*

   Replace "A *literal* represents" with "A *literal* is represented by". [Note: Typo.]

4. *Page 3, 5th paragraph*

   Replace "whose values obey" with "whose arguments obey". [Note: These are conventions for arguments.]

5. *Page 4, section 1.4, 3rd paragraph*

   Replace "see §13.1 and §8" with "see §8 and §13.1". [Note: Change the order.]

6. *Page 7, clause 1.7.22*

   Replace "ISLISP supports" with "ISLisp supports". [Note: Change the font.]

7. *Page 7, clause 1.7.22*

   Replace "several superclasses" with "several direct superclasses".

8. *Page 8, clause 1.7.28*

   Replace "the first element" with "The first element". [Note: Capitalize.]

9. *Page 9, section 1.8.1, item (a), 2nd paragraph*

   Replace "the current expression" with "the current form". [Note: The previous paragraph uses "form" rather than "expression".]

10. *Page 9, section 1.8.1, item (a), 2nd paragraph*

    Replace "It is implementation defined whether" with "If no active handler is established by `with-handler`, it is implementation defined whether". [Note: If an appropriate handler is established, the behavior is well defined.]

11. *Page 11, 1st paragraph*

    Replace "by indenting C_2 under C_1 in Figure 1" with "by use of a directed arrow from C_1 to C_2 in Figure 1". [Note: The IS uses both "indentation" and "arrow" to express class relations. Use either one. Japanese Lisp WG prefers "arrow" because it is clearer than "indentation".]

12. *Page 11, 5th paragraph*

    Replace “there is an edge from C_1 to C_2 iff C_1 is direct subclass of C_2” with “there is an edge from C_1 to C_2 iff C_2 is direct subclass of C_1”. [Note: Exchange “C_1” and “C_2”.]

13. *Page 12, Figure 1*

    Replace the entire figure with Figure 1 of this corrigendum. [Note: See the Note for Item 11 above.]

14. *Page 13, section 2.2, 3rd paragraph*

    Replace “as an instance of `<built-in-class>` or as an instance of `<built-in-class>`” with “as an instance of `<built-in-class>`”. [Note: Duplicated.]

15. *Page 14, section 3, 1st paragraph*

    Replace “an ISLisp text (see §1.3) within” with “an ISLisp text (see §1.3 and §1.7.37) within”. [Note: §1.7.37 is also a good cross-reference.]

16. *Page 15, section 3.3, 1st paragraph*

    Replace “required built-in functions, required built-in macros, and constants” with “required built-in functions, special operators, defining operators, and constants”. [Note: What are called built-in macros in other languages are called special operators or defining operators in ISLisp.]

17. *Page 16, section 3.4*

    Add `dynamic-let` in the list of binding forms. The list should look as follows.

    ```
    block        let                   with-open-io-file
    dynamic-let  let*                  with-open-output-file
    flet         tagbody               with-standard-input
    for          with-error-output     with-standard-output
    labels       with-open-input-file
    ```

18. *Page 17, section 4.1, 2nd paragraph from bottom*

    Replace “The *operator* must be a special operator, or an identifier or a lambda expression” with “The *operator* must be a special operator, a defining operator, an identifier, or a lambda expression”. [Note: The *operator* may be a defining operator.]

19. *Page 17, section 4.1, 2nd paragraph from bottom*

    Replace “The identifier names a function, or a generic function” with “The identifier names a function, a macro, or a generic function”. [Note: The identifier may name a macro.]

20. *Page 18, section 4.3*

    Add `ignore-errors` and `set-dynamic` in the list of special operators. The list should look as follows.

    ```
    and         dynamic-let    let*          throw
    assure      flet           or            unwind-protect
    block       for            progn         while
    case        function       quote         with-error-output
    case-using  go             return-from   with-handler
    catch       if             set-dynamic   with-open-input-file
    class       ignore-errors  setf          with-open-io-file
    cond        labels         setq          with-open-output-file
    convert     lambda         tagbody       with-standard-input
    dynamic     let            the           with-standard-output
    ```

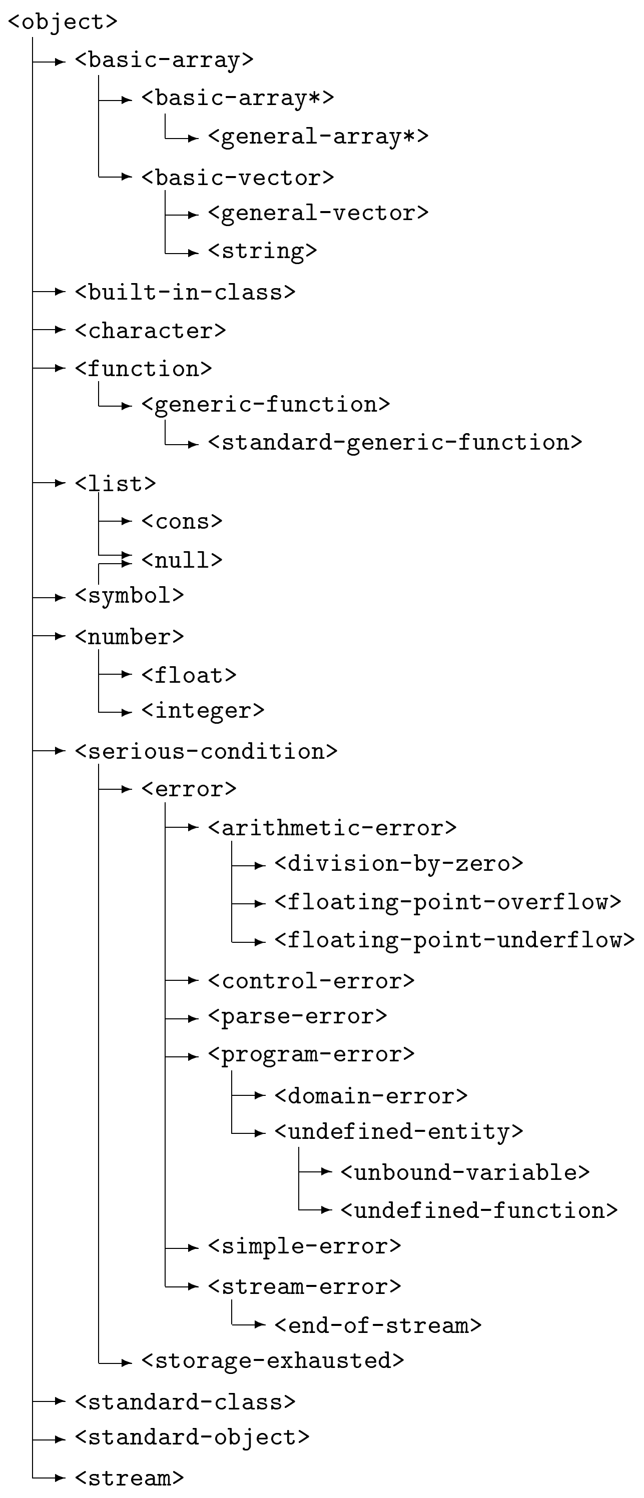

Figure 1: Class Inheritance

21. *Page 19, section 4.5*

    Delete the sentence: “It is implementation defined whether any operator described by this document as a macro is implemented as a special operator (see §4.3).” [Note: No operator is described as a macro.]

22. *Page 20, item (d)*

    Replace “The *arguments* are evaluated in order from left to right, yielding objects (sometimes called “actual arguments”) to which the function will be applied” with “The *arguments* are evaluated in order from left to right”. [Note: The second half “yielding …” is unnecessary. In fact, the similar description in item (c) does not contain it. In addition, the term “actual arguments” is never used in the IS.]

23. *Page 21, definition of* `function` *etc*

    Replace “named by the identifier *function-name*” with “named by *function-name*”. [Note: The *function-name* may be a lambda list.]

24. *Page 22, definition of* `labels` *etc, the header*

    Replace two occurrences of “*body-forms**” with “*body-form**”. [Note: It is a sequence of *body-form*.] The header should look as follows.

    `(labels ((`*function-name* *lambda-list* *form*`*)*)` *body-form*`*)` → <*object*> **special operator**
    `(flet ((`*function-name* *lambda-list* *form*`*)*)` *body-form*`*)` → <*object*> **special operator**

25. *Page 22, definition of* `labels` *etc, items list*

    In the first item, replace “bindings for the given *funs*” with “bindings for the given *function-names*”. [Note: “*function-name*” is used in the header.]

26. *Page 22, definition of* `labels` *etc, 3rd paragraph*

    Replace “by the function activation” with “by the special form activation”. [Note: `labels` and `flet` are special forms, but not functions.]

27. *Page 23, definition of* `funcall`*, 2nd paragraph*

    Replace “Each *argument* may be any ISLISP object” with “Each *obj* may be any ISLISP object”. [Note: “*obj*” is used in the header.]

28. *Page 25, definition of* `defdynamic`*,* Example

    Replace “`red`” with “`*color*`” as the value of the first form “`(defdynamic *color* 'red)`”. [Note: `defdynamic` returns the name of the variable.]

29. *Page 25, definition of* `defun`*, 3rd paragraph*

    Replace “The free identifiers in the body” with “The free identifiers in the body *form**”. [Note: The “body” of `defun` is not defined before.]

30. *Page 26, definition of* `eq` *etc*

    Replace “if the *objects* are the same” with “if the objects are the same”. [Note: Change the font.]

31. *Page 27, 1st paragraph*

    Replace “the consequences are undefined if either obj_1 or obj_2 is a number or a character” with “the consequences are implementation defined if both obj_1 and obj_2 are numbers or both are characters”. [Note: “undefined” is an extreme. In addition, the original phrase after “if” is ambiguous.]

32. *Page 27, items list*

    In the first item, replace "of the same classes" with "of the same class". [Note: Make it singular.]

33. *Page 28, definition of* `equal`*, 2nd paragraph*

    Replace "If *obj*$_1$ and *obj*$_2$ are instances of the same classes" with "If *obj*$_1$ and *obj*$_2$ are direct instances of the same class". [Note: Since any object is an instance of `<object>`, this condition does not make sense without "direct". Also, make "classes" sigular.]

34. *Page 28, definition of* `equal`*, 2nd paragraph*

    Replace "of the same classes but not `eql`" with "of the same class but not `eql`". [Note: Make it singular.]

35. *Page 31, section 6.2, 1st paragraph*

    Replace "***Variable bindings***, or *variables*, are entities" with "***Variable bindings*** are entities". [Note: "Variable bindings" and "variables" are different concepts. See section 1.7.4 in Page 6. Also, see Item 36 below.]

36. *Page 31, section 6.2, 2nd paragraph*

    Replace "A ***variable*** is an association between an *identifier* and an ISLisp object and is denoted by that identifier" with "A ***variable*** is used to refer to an association between an *identifier* and an ISLisp object, and is denoted by that identifier". [Note: The definition of "variable" in the IS should be replaced with a widely acceptable one such as the one given here.]

37. *Page 33, list of valid places for* `setf`

    Replace "(*accessor-name instance*)" with "(*reader-function-name instance*)". [Note: See the syntax definition of `defclass` in Page 46.]

38. *Page 33, 1st paragraph after list of valid places for* `setf`

    Replace "the arguments of the *place* form" with "the arguments of the function application form". [Note: This sentence is talking about function application forms, but not general places.]

39. *Page 33,* Example

    Replace two occurrences of "`car`" in comments with "*car*", and adjust indentation of the line with the second `setf` form. [Note: The "car" here is not a function name and therefore should be in Roman face if it appeared in ordinary text. Since the comments are written in Italic face, the "car" here should be in Italic face.] The example should look as follows.

```
(setf (car x) 2)                ⇒  2
 In the cons x, the car now is 2.

(defmacro first (spot)
  `(car ,spot))                 ⇒ first
(setf (first x) 2)              ⇒  2
 In the cons x, the car now is 2.
```

40. *Page 33, definition of* `let`*, 2nd paragraph*

    Replace "The forms *form* are evaluated" with "The *forms* are evaluated".

41. *Page 33, definition of* `let`*, 2nd paragraph*

    Replace “the *forms* are evaluated” with “the *body-forms* are evaluated”. [Note: “*body-form**” is used in the header and the IS sometimes refers to “*X**” as “*Xs*”.]

42. *Page 33, definition of* `let`*,* **Note**

    Add a missing closing parenthesis for the second `let` form. The Note should look as follows.

```
(let () body-form*)           ≡     (progn body-form*)[3]
(let ((var1 form1)            ≡     ((lambda (var1 var2 ... varn)
      (var2 form2)                        body-form*
         ...                         ) form1 form2 ... formn)[4]
      (varn formn))
  body-form*)
```

43. *Page 34, definition of* `let*`*, 1st paragraph*

    Replace “The scope of an identifier *var* is the *body* excluding nested regions of *var*, if any, along with” with “The scope of an identifier *var* is the *body* along with”. [Note: Use the simpler definition of “scope”, which is used in the definition of `let`.]

44. *Page 34, definition of* `let*`*, 2nd paragraph*

    Replace “These definitions enlarge the set of current valid identifiers perhaps shadowing previous definitions” with “These variable bindings enlarge the set of current valid identifiers perhaps shadowing previous variable bindings”. [Note: “Definition” here does not make sense.]

45. *Page 34, definition of* `let*`*, 2nd paragraph*

    Replace “in this enlarged or modified environment the *forms* are executed” with “in this enlarged or modified environment the *body-forms* are executed”. [Note: “*body-form**” is used in the header and the IS sometimes refers to “*X**” as “*Xs*”.]

46. *Page 34, definition of* `let*`*,* **Note**

    In the second equivalence relation, replace the four closing parentheses in Roman face with those in Type face. Also, replace “...” with “. . .”. [Note: The spaces between dots are different.] The second equivalence relation should look as follows.

```
(let* ((var1 form1)    ≡  (let ((var1 form1))
       (var2 form2)          (let ((var2 form2))
          ...               ...
       (varn formn))           (let ((varn formn))
   body-form*)                     body-form*)...))
```

47. *Page 34, footnote 4*

    Replace “see 5.6.d” with “see §4.7”. [Note: There is no reference point that corresponds to “5.6.d”.]

48. *Page 35, definition of* `(setf (dynamic ...) ...)`*, the header*

    Add `set-dynamic`. The header should look as follows.

`(setf (dynamic` *var*`)` *form*`)` → <*object*>	**special form**
`(set-dynamic` *form* *var*`)` → <*object*>	**special operator**

49. *Page 37, equivalence relations for* `cond`

    Enlarge asterisks. The equivalence relations should look as follows.

```
(cond)                         ≡    nil
(cond ($test_1$)               ≡    (or $test_1$
      ($test_2$ $form_2^*$)             (cond ($test_2$ $form_2^*$)
         ...)                                 ...))
(cond ($test_1$ $form_1^+$)    ≡    (if $test_1$
      ($test_2$ $form_2^*$)             (progn $form_1^+$)
         ...)                           (cond ($test_2$ $form_2^*$)
                                              ...))
```

50. *Page 39, definition of* `for`*, 4th paragraph*

    Replace “The `for` macro is executed” with “The `for` special form is executed”. [Note: `for` is a special form, but not a macro.]

51. *Page 40, section 6.7.1, 1st paragraph*

    In the table, replace “*block tag*” with “*block name*”. [Note: The IS uses the term “name” rather than “tag” for blocks. See the definitions of `block` and `return-from`.]

52. *Page 42, definition of* `catch` *etc, 1st paragraph*

    Replace “A catch tag may be any object other than a number or a character” with “A catch tag may be any object other than a number or a character; the comparison of *catch tags* uses `eq`”. [Note: The description in the fifth paragraph in the definition of `catch` etc is duplicated. Delete the entire paragraph and merge the contents to the first paragraph.]

53. *Page 42, definition of* `catch` *etc, 2nd paragraph*

    Replace “produce a ***catch tag***” with “produce a *catch tag*”. [Note: Change the font. “***catch tag***” is already defined in the first paragraph.]

54. *Page 42, definition of* `catch` *etc, 4th paragraph*

    Replace three occurrences of “C_i” with “C_1”. Also, replace “R_i” with “R_1”. [Note: There is no reason for using general subscript “i” here.]

55. *Page 42, definition of* `catch` *etc, 5th paragraph*

    Delete the entire paragraph. [Note: See Item 52 above.]

56. *Page 43, definition of* `tagbody` *etc, 2nd paragraph*

    Add a period at the end of the paragraph.

57. *Page 43, definition of* `tagbody` *etc, 4th paragraph*

    Replace “a form *(go* tag_i*)* can be” with “a form `(go` *tag*`)` can be”. [Note: Change the font. In addition, there is no reason for using general subscript “i” here.]
    Also, replace the other two occurrences of “tag_i” with “*tag*”.

58. *Page 43, definition of* `tagbody` *etc,* **Note**

    Replace “that parallels the unstructured imperative transfer of control” with “that uses unstructured imperative transfer of control”. [Note: “parallels” may cause confusion.]

59. *Page 43, definition of* `tagbody` *etc,* **Note**

    Delete “that these facilities provide”. [Note: “these facilities” does not make sense.]

60. *Page 44, definition of* `unwind-protect`*,* **Note**

    Replace "respect these `cleanup-forms`" with "respect these *cleanup-forms*". [Note: Change the font.]

61. *Page 46, definition of* `defclass`*, the syntax table*

    Add a line "*boundp-function-name* ::= *identifier*" between syntax definitions for *writer-function-name* and *class-opt*.

62. *Page 49, section 7.1.3, 3rd paragraph, items list*

    In the first item, replace "*C*'s class precedence list of the classes that define them" with "*C*'s class precedence list". [Note: Delete the redundant clause.]

63. *Page 50, 4th paragraph*

    Replace "ISLisp provides a default method combination type and provides a facility for declaring new types of method combination" with "ISLisp provides a default method combination type but does not provide a facility for declaring new types of method combination". [Note: ISLisp does not provide such a facility.]

64. *Page 50, section 7.2.1, 1st paragraph*

    Delete "or its argument precedence order". [Note: There is no way to specify argument precedence order.]

65. *Page 50, definition of* `defgeneric`*, the syntax table*

    Replace the first line of the syntax definition for *option* with "*option* ::= `(:method-combination` {*identifier* | *keyword*}`)` |". [Note: "Symbol" should be "identifier" according to the convention of the IS (see, for example, the syntax definition of `defclass` in Page 46). Also, keywords can be specified as method combination (see the second item of the items list in Page 51).]

66. *Page 51, definition of* `defgeneric`*, the syntax table*

    Replace the syntax definition for *method-qualifier* with "*method-qualifier* ::= *identifier* | *keyword*". [Note: Symbol can be used as a method qualifier in `defmethod` (see the fifth paragraph from bottom of Page 52) and *method-qualifier* in `defgeneric` is the same as that in `defmethod` (see the paragraph after the items list in Page 51).]

67. *Page 51, definition of* `defgeneric`*, the syntax table*

    Add a line "*var* ::= *identifier*" at the end of the syntax table.

68. *Page 51, definition of* `defgeneric`*, 5th paragraph*

    Delete the entire paragraph "Each *method-desc* defines . . . in this context". [Note: The same description is given later.]

69. *Page 51, definition of* `defgeneric`*, 2nd paragraph before section 7.2.2*

    Replace "All methods on the resulting generic function must have lambda-lists that are congruent with this shape" with "All methods on the resulting generic function must have parameter-profiles that are congruent with this shape". [Note: Methods are given parameter-profiles, but not lambda-lists.]

70. *Page 52, definition of* `defmethod`*, the syntax table*

    Replace the syntax definition for *method-qualifier* with "*method-qualifier* ::= *identifier* | *keyword*". [Note: Symbol can be used as a method qualifier in `defmethod` (see the fifth paragraph from bottom of Page 52).]

71. *Page 52, definition of* `defmethod`*, the syntax table*

    In the syntax definition for *parameter-profile*, replace “var” with “*var*”. [Note: Change the font.]

72. *Page 52, definition of* `defmethod`*, the syntax table*

    In the syntax definition for *parameter-profile*, replace the four parentheses in Roman face with those in Type face.

73. *Page 52, definition of* `defmethod`*, the syntax table*

    Add a line “*var* ::= *identifier*” at the end of the syntax table.

74. *Page 52, 6th paragraph from bottom*

    Replace “The lambda-list of the method being defined” with “The parameter-profile of the method being defined”. [Note: Methods are given parameter-profiles, but not lambda-lists.]

75. *Page 53, 1st paragraph*

    Delete the entire paragraph “Each method has ... required parameter”. [Note: The terms defined here are never used.]

76. *Page 53, section 7.2.2.1, 2nd paragraph*

    Replace “derived from the parameter specializer names as described above” with “derived from the parameter profiles as described above”. [Note: “parameter profiles” is more precise. See the fourth paragraph from bottom in Page 52.]

77. *Page 53, section 7.2.2.2, 1st paragraph*

    Replace “including the lambda-list of each method” with “including the parameter profile of each method”. [Note: Methods are given parameter-profiles, but not lambda-lists.]

78. *Page 53, section 7.2.3, 2nd paragraph*

    Replace “by using one of the method-defining forms” with “by using the `defmethod` form”. [Note: ISLisp defines only `defmethod` as method-defining forms.]

79. *Page 54, section 7.3.1, 2nd paragraph*

    Replace “If P_i is a class, and if A_i is an instance” with “If A_i is an instance”. [Note: P_i is always a class.]

80. *Page 54, section 7.3.1, 4th paragraph*

    Replace the entire sentence “A qualifier is any object other than a list; *i.e.,* any non-`nil` symbol or keyword” with “Any object after `:method` and before the first list in *method-desc* is regarded as a qualifier, but only non-`nil` symbols and keywords are accepted as qualifiers”. [Note: Non-`nil` symbols and keywords are not the only objects other than lists.]

81. *Page 55, section 7.3.3.1*

    Replace “the effective method is the most specific method” with “the effective method calls the most specific method”. [Note: Typo.]

82. *Page 56, items list*

    In the last item, replace “The value returned by the invocation of `call-next-method` in the least specific `:around` method are those returned by” with “The value returned by the invocation of `call-next-method` in the least specific `:around` method is that returned by”. [Note: Editorial error.]

83. *Page 57, 2nd paragraph*

    Replace “the method combination qualifier is `nil`” with “the method combination type is `nil`”. [Note: Typo. See the definition of `defgeneric` in Page 51.]

84. *Page 57, 3rd paragraph*

    Replace the sentences “The standard method combination type defines the next method as follows ... see §7.3.3” with “The standard method combination type defines the next method as specified in §7.3.3.2”. [Note: The descriptions are not precise. §7.3.3.2 gives precise descriptions.]

85. *Page 58, 2nd paragraph*

    Delete the entire paragraph including items list “ISLisp specifies system-supplied primary methods for each step ... a system-supplied primary method for `initialize-object`”. [Note: The contents of this paragraph are given in Page 47 and in section 7.4.1 of Page 58. In addition, “each step” in the paragraph is misleading because the system-supplied primary method does not handle the first step “allocating storage for the instance”.]

86. *Page 58, last paragraph*

    Delete the entire paragraph “Methods for `initialize-object` can be ... behavior of `initialize-object`”. [Note: This paragraph is duplicated with the third paragraph of the definition of `initialize-object` in Page 59.]

87. *Page 59, definition of* `initialize-object`*, 2nd paragraph*

    Replace “the `:initform` forms of the slots (see §7.4.1)” with “the `:initform` forms of the slots”. [Note: The sentence itself is in §7.4.1.]

88. *Page 59, definition of* `class-of`

    Replace three occurrences of “*object*” with “*obj*”. [Note: Typo. See Page 3.]

89. *Page 59, definition of* `instancep`

    Replace three occurrences of “*object*” with “*obj*”. [Note: Typo. See Page 3.]

90. *Page 62, 1st paragraph*

    Replace “In a `the` special form, the consequences are undefined if the value of *form* is not of the class or a subclass of the class designated by *class-name* (error-id. *domain-error*)” with “In a `the` special form, the consequences are undefined if the value of *form* is not of the class or a subclass of the class designated by *class-name*”. [Note: No error-id should be specified if the consequences are undefined. In addition, the example in the definition says “*the consequences are undefined*”.]

91. *Page 63, items list*

    In the second item, replace “using the following: `(create-string 1 obj)`” with “using the following: `(create-string 1` *obj*`)`”. [Note: Change the font.]

92. *Page 64, paragraph right before section 10.1.2*

    Replace “`&rest`, `:rest`, and keywords (*e.g.,* `:before` and `:after`)” with “`&rest` and keywords (*e.g.,* `:rest`, `:before`, and `:after`)” [Note: `:rest` is a keyword.]

93. *Page 69, 1st paragraph*

    Replace “by `(convert <float>` *z*`)`” with “by `(convert` *z* `<float>)`”. [Note: Change the argument order.]

94. *Page 72, definition of* `sqrt`*, 1st paragraph*

    Replace “Returns the square root” with “Returns the non-negative square root”. [Note: There are two square roots for each positive number.]

95. *Page 73, definition of* `atan`*, 1st paragraph*

    Delete the sentences “This can be mathematically defined as follows . . . for real-valued computations”. [Note: ISLISP does not support complex numbers.]

96. *Page 73,* **Note** *at the bottom*

    Delete the entire **Note** “Beware of simplifying this formula . . . is strictly negative”. [Note: ISLISP does not support complex numbers.]

97. *Page 74, Figure 3*

    Replace two occurrences of “$+\pi$” with “π”. [Note: Other positive numbers in the table are not prefixed with “+”.]

98. *Page 74, Figure 3*

    Replace “undefined consequences” with “implementation defined”. [Note: See the definition of `atan2`.]

99. *Page 74, definition of* `atan2`*, 4th paragraph*

    Replace “The signs of x_1 (indicated as $\underline{x}$) and x_2 (indicated as $\underline{y}$) are used” with “The signs of x_1 (indicated as $\underline{y}$) and x_2 (indicated as $\underline{x}$) are used”. [Note: Exchange x and y.]

100. *Page 75, last paragraph*

     Delete the entire paragraph “The following definition for . . . if its imaginary part is strictly positive”. [Note: ISLISP does not support complex numbers.]

101. *Page 79, last line*

     Replace “`(=` $\underline{z_2}$ `(+ (* (div` z_1 z_2`)` z_2`) (mod` z_1 z_2`)))`” with “`(=` $\underline{z_1}$ `(+ (* (div` z_1 z_2`)` z_2`) (mod` z_1 z_2`)))`”. [Note: Typo.]

102. *Page 80, definition of* `gcd`*, 1st paragraph*

     Replace “the largest integer such z that” with “the largest integer z such that”. [Note: Typo.]

103. *Page 81, chapter 12, 1st paragraph*

     Replace the comma at the end of the paragraph with a period. [Note: Typo.]

104. *Page 83, section 13.1, 1st paragraph*

     Replace “the left component is called `car` and the right component is called `cdr`” with “the left component is called car and the right component is called cdr”. [Note: Change the font. The IS uses Roman font for car and cdr in other places.]

105. *Page 83, section 13.1, 1st paragraph*

     Replace “denote the values in the `car` and `cdr` components” with “denote the values in the car and cdr components”. [Note: Change the font. The IS uses Roman font for car and cdr in other places.]

106. *Page 83, section 13.1, 1st paragraph*

     Replace “if the `cdr` value is `nil`” with “if the *cdr* value is `nil`”. [Note: Change the font. “cdr” here denotes the object in the cdr component.]

107. *Page 85, definition of* `set-car` *etc, 1st paragraph*

     Replace “The `setf` special form takes the place indicated by the selector `car` and updates the left component of an instance of the `<cons>`” with “Updates the left component of *cons* with *obj*”. [Note: Define `set-car` as well. In addition, specify the new value for the place.]

108. *Page 85, definition of* `set-car` *etc, 1st paragraph*

Replace "The returned value is the result of the evaluation of *obj*" with "The returned value is *obj*". [Note: *obj* is already evaluated.]

109. *Page 85, definition of* `set-cdr` *etc, 1st paragraph*

Replace "The `setf` special form takes the place indicated by the selector `cdr` and updates the right component of an instance of `<cons>`" with "Updates the right component of *cons* with *obj*". [Note: Define `set-cdr` as well. In addition, specify the new value for the place.]

110. *Page 85, definition of* `set-cdr` *etc, 1st paragraph*

Replace "The returned value is the result of the evaluation of *obj*" with "The returned value is *obj*". [Note: *obj* is already evaluated.]

111. *Page 86, definition of* `create-list`*, 1st paragraph*

Replace "An error shall be signaled if i is not an integer (error-id. *domain-error*)" with "An error shall be signaled if i is not a non-negative integer (error-id. *domain-error*)".

112. *Page 87, definition of* `reverse` *etc,* Example

Remove the blank line right before the last line.

113. *Page 90, definition of* `assoc`*, the header*

Replace "<*cons*>" with "<*list*>". [Note: `assoc` may return `nil`.] The header should look as follows.

`(assoc` *obj association-list*`)` $\rightarrow$ <*list*> **function**

---

114. *Page 90, explanation of* `<basic-array>`*, last line*

Replace "subclasses of of `<basic-array>`" with "subclasses of `<basic-array>`". [Note: Typo.]

115. *Page 92, definition of* `create-array`*,* Example

Replace "`(create-array '(2) 0.)`" with "`(create-array '(2) 0.0)`".

116. *Page 93, 1st paragraph*

Replace "d_i the i$^{\text{th}}$ dimension" with "d_i the i^{th} dimension". [Note: Change the font.]

117. *Page 93, definition of* `set-aref` *etc, 1st paragraph*

Replace "With `setf` the object obtainable by `aref` or `garef`, respectively, is replaced" with "These replace the object obtainable by `aref` or `garef` with *obj*. The returned value is *obj*". [Note: Define `set-aref` as well. In addition, specify the new value for the place and the returned value.]

118. *Page 95, chapter 16, 2nd paragraph*

Replace "representation of non-printable characters" with "representation of non-printing characters". [Note: Typo. See section 12 in Page 81.]

119. *Page 97, definition of* `char-index`*, the header*

Replace "*character*" with "*char*". [Note: Typo. See Page 3.] The header should look as follows.

`(char-index` *char string* [*start-position*]`)` $\rightarrow$ <*object*> **function**

---

120. *Page 97, definition of* `char-index`*, 1st and 2nd paragraphs*

Replace three occurrences of "*character*" with "*char*". [Note: Make them compatible with the corrected header. See Item 119 above.]

121. *Page 97, definition of* `char-index`*, 1st paragraph*

Replace "The function `eql` is used" with "The function `char=` is used". [Note: `char=` is more specific.]

122. *Page 97, definition of* `string-index`*, 1st paragraph*

Replace "sequential use of `eql` on" with "sequential use of `char=` on". [Note: `char=` is more specific.]

123. *Page 98, definition of* `length`*, 3rd paragraph*

Replace "Consistently with that, `'(a b . c)` ≡ `(cons 'a (cons 'b 'c))` and `(length '(a b . c))` ⇒ 2" with "For example, `(length '(a b . c))` ⇒ 2, since `'(a b . c)` ≡ `(cons 'a (cons 'b 'c))`". [Note: "`'(a b . c)` ≡ `(cons 'a (cons 'b 'c))`" is the reason for "`(length '(a b . c))` ⇒ 2".]

124. *Page 99, definition of* `elt`*, the header*

Delete the blank line in the header. The header should look as follows.

`(elt` *sequence z*`)` → <*object*> **function**

---

125. *Page 99, definition of* `set-elt` *etc, 1st paragraph*

Replace "The `setf` special form takes the place and updates this place with the result of the evaluation of *obj*" with "These replace the object obtainable by `elt` with *obj*. The returned value is *obj*". [Note: Define `set-elt` as well. In addition, specify the new value for the place. Finally, *obj* is already evaluated.]

126. *Page 99, definition of* `set-elt` *etc, 1st paragraph*

Delete the sentence "The integer z satisfies $0 \le z <$ `(length` *sequence*`)`". [Note: This condition is mentioned in the second paragraph.]

127. *Page 99, definition of* `set-elt` *etc, 2nd paragraph*

Delete the sentence "The returned value is the result of the evaluation of *obj*". [Note: See Item 125 above.]

128. *Page 100, definition of* `map-into`*, the header*

Replace "*seq*" with "*sequence*". [Note: Typo. See Page 3.] The header should look as follows.

`(map-into` *destination function sequence**`)` → *sequence* **function**

---

129. *Page 100, definition of* `map-into`*, 1st, 2nd, and 4th paragraphs*

Replace four occurrences of "*seqs*" with "*sequences*". [Note: Make them compatible with the corrected header. See Item 128 above.]

130. *Page 102, definition of* `with-standard-input` *etc, 1st paragraph*

Replace "These macros first evaluate" with "These special forms first evaluate". [Note: `with-standard-input` etc are special forms, but not macros.]

131. *Page 102, definition of* `with-standard-input` *etc, 1st paragraph*

Replace "return the stream *s*" with "returns the stream *s*". [Note: Typo.]

132. *Page 102, definition of* `with-standard-input` *etc, 1st paragraph*

Add "The returned value of each of these forms is the result of the evaluation of the last *form* of their body (or `nil` if there is none)" at the end of the paragraph.

133. *Page 103, definition of* `with-open-input-file` *etc, the header*

Replace "*file*" with "*filename*". [Note: Make them compatible with the headers for `open-input-file` etc.] The header should look as follows.

`(with-open-input-file (`*name* *filename* [*element-class*]`)` *form**`)` → <*object*> **special operator**

`(with-open-output-file (`*name* *filename* [*element-class*]`)` *form**`)` → <*object*> **special operator**

`(with-open-io-file (`*name* *filename* [*element-class*]`)` *form**`)` → <*object*> **special operator**

---

134. *Page 103, definition of* `with-open-input-file` *etc, 1st paragraph*

Replace "Each of these macros opens" with "Each of these special forms opens". [Note: `with-open-input-file` etc are special forms, not macros.]

135. *Page 103, definition of* `with-open-input-file` *etc, 1st paragraph*

Replace "respectively). evaluates the" with "respectively), evaluates the". [Note: Replace the period with a comma.]

136. *Page 103, definition of* `with-open-input-file` *etc, 2nd paragraph*

Replace "*file*" with "*filename*". [Note: Make them compatible with the corrected header. See Item 133 above.]

137. *Page 103, definition of* `with-open-input-file` *etc, 2nd paragraph*

Delete the sentences "Then the *forms* are ... is returned". [Note: Mentioned already in the first paragraph.]

138. *Page 103, definition of* `with-open-input-file` *etc, 3rd paragraph*

Replace "from this macro is normal. For this reason, these macros are usually" with "from these special forms is normal. For this reason, these special forms are usually". [Note: `with-open-input-file` etc are special forms, but not macros.]

139. *Page 105, definition of* `get-output-stream-string`, Example

Replace "`(get-output-string-stream)`" with "`(get-output-stream-string out-str)`". [Note: Correct the function name and give an argument.]

140. *Page 108, definition of* `format` *etc, 1st paragraph*

Insert "It returns `nil`." after the first sentence. [Note: The examples say `format` returns `nil`.]

141. *Page 110, section 19.3, 1st paragraph*

Replace "perform a character I/O operation" with "perform a binary I/O operation". [Note: This section is about binary I/O.]

142. *Page 110, definition of* `read-byte`, Example

Replace "`nil`" with "*implementation-defined*" as the value of the third form "`(close byte-example)`". [Note: See the definition and examples of `close` in Page 103.]

143. *Page 110, definition of* `read-byte`*,* Example

    Replace "`97`" with "`101`" as the value of the sixth form "`(read-byte byte-example)`". [Note: The ASCII code for "`e`" is 101.]

144. *Page 111, definition of* `write-byte`*,* Example

    Replace "`nil`" with "*implementation-defined*" as the value of the form. [Note: See the definition and examples of `close` in Page 103.]

145. *Page 111, definition of* `probe-file`*,* Example

    Replace "`nil`" with "*implementation-defined*" as the value of the third form "`(close new-file)`". [Note: See the definition and examples of `close` in Page 103.]

146. *Page 111, definition of* `file-position`*, 2nd paragraph*

    Replace "increased by one each time a one of" with "increased by one each time one of". [Note: Typo.]

147. *Page 112, definition of* `file-position`

    In the second list of function calls, add a line "`(report-condition` *condition stream*`)`". [Note: `report-condition` also outputs characters.]

148. *Page 112, 2nd paragraph*

    Replace "If a stream supports file positions, it is implementation defined which integer" with "It is implementation-defined which integer". [Note: The condition is mentioned in the next sentence.]

149. *Page 112, definition of* `file-position`*,* Example

    Replace "`nil`" with "*implementation-defined*" as the value of the third form "`(close example)`". [Note: See the definition and examples of `close` in Page 103.]

150. *Page 112, definition of* `file-position`*,* Example

    Replace "`0`" with "`0` *(implementation-defined)*" as the value of the fifth form "`(file-position example)`". [Note: See the definition of `file-position` in Page 111.]

151. *Page 112, definition of* `file-position`*,* Example

    Replace "`1`" with "`1` *(implementation-defined)*" as the value of the seventh form "`(file-position example)`". [Note: See the definition of `file-position` in Page 111.]

152. *Page 113, definition of* `file-length`*, the header*

    Replace "<*integer*>" with "<*object*>". [Note: `file-length` may return `nil`.] The header should look as follows.

    `(file-length` *filename element-class*`)` → <*object*> **function**

153. *Page 113, definition of* `file-length`*, 1st paragraph*

    Replace "if *filname* is not" with "if *filename* is not". [Note: Typo.]

154. *Page 113, section 21.1, 4th paragraph*

    Replace "Conditions that represent implementation limitations that may not be symptomatic of program errors are called ***serious conditions***" with "Error conditions and those conditions that represent implementation limitations that may not be symptomatic of program errors are collectively called ***serious conditions***". [Note: Serious conditions include error conditions, see Figure 1 in Page 12.]

155. *Page 114, section 21.2, 5th paragraph*

 Replace "it must ***handle*** the condition" with "it must handle the condition". [Note: Change the font. "handle" here is a general verb.]

156. *Page 115, definition of* `cerror`

 In the equivalent code, delete the last closing parenthesis of the fourth line. [Note: Typo.]

157. *Page 115, definition of* `cerror`

 In the equivalent code, replace "`(create-string-output-string)`" with "`(create-string-output-stream)`". [Note: Typo.]

158. *Page 118, section 21.3.6*

 In the specification of `<undefined-entity>`, replace "a symbol representing of the identifier" with "a symbol representing the identifier". [Note: Typo.]

159. *Page 120, specification for unbound-variable*

 Replace "is made to refer to an unbound" with "is made to access an unbound". [Note: Assignment may also cause this error.]

160. *Page 120, specification for undefined-entity*

 Replace "when a reference to that entity is made" with "when an access to that entity is made". [Note: Assignment may also cause this error.]

161. *Page 121, definition of* `get-internal-run-time` *etc, 1st paragraph*

 Replace "returns as an *integer* the" with "returns as an integer the". [Note: Change the font. There is no reason to use Italic font here.]

162. *Index*

 Delete the entry for "`<simple-error>`". [Note: No other class names are included in the Index.]

163. *Index*

 Merge duplicated entries for the following symbols in Roman font.

 array, character, cons, float, integer, list, null, string, symbol, vector

164. *Index*

 Replace "`create-string-output-string`" with "`create-string-output-stream`" and merge the entry with that of `create-string-output-stream`. [Note: The typo mentioned in Item 157 caused this error.]

165. *Index*

 Add an entry for "writer (of a slot)". [Note: The Index contains entries for "reader (of a slot)" and "accessor (of a slot)".]

166. *Index*

 Add an entry for "`set-dynamic`".

167. *Index*

 Delete the entry for "handle". [Note: See Item 155.]

168. *Index*

 Delete the entries for "specialized lambda-list" and "specialized parameter". [Note: These appear only in the paragraph to be deleted. See Item 75.]