Computer Programs as Literary Works and as Modes of Operation: A Case Comment on Lotus v. Borland

Jeffrey Brill
COMPUTER PROGRAMS AS LITERARY WORKS AND AS MODES OF OPERATION: A CASE COMMENT ON LOTUS V. BORLAND

JEFFREY BRILL*

INTRODUCTION

The distinction between the copying of literal and nonliteral elements of computer programs raises several interesting and, as yet, not clearly settled copyright infringement questions. Congress has legislated that computer programs constitute literary works and, as such, are the subject of copyright protection.1 As courts have held,2 a line by line copying of the source or object code of a computer program clearly constitutes copyright infringement. At the same time, the Copyright Act stipulates that no copyright protection exists for "any idea, procedure, process, system, method of operation, [or] concept."3 While computer programs often have creative elements, they are obviously more utilitarian in nature than traditional literary works. Since perhaps every nonliteral element of a computer program constitutes an "idea, procedure, process, system, method of operation, [or] concept," the appropriate amount of copyright protection to extend to the copying of nonliteral elements of computer programs is unclear. The holding in Lotus Development Corp. v. Borland International, Inc.4 indicates that perhaps the correct

* Juris Doctor, Chicago-Kent College of Law, Illinois Institute of Technology, 1998. The author is an associate at Fenwick & West LLP in Palo Alto, California. The author wishes to thank Professor Mickie Voges for reviewing and commenting on this work.

1. See 17 U.S.C. § 102(a)(1) (1994) (instructing that literary works are the subject of copyright protection); id. § 101 (defining "literary works" as "works . . . expressed in words, numbers, or other verbal or numerical symbols or indicia, regardless of the nature of the material objects, such as books . . . tapes, disks . . . in which they are embodied"); see also H.R. Rep. No. 94-1476, at 54 (1976), reprinted in 1976 U.S.C.C.A.N. 5659, 5667 ("The term 'literary works' . . . includes . . . computer programs to the extent that they incorporate authorship in the programmer's expression of original ideas, as distinguished from the ideas themselves.").

2. See CCMS Software Design Sys., Inc. v. Info Designs, Inc., 785 F.2d 1246, 1247 (5th Cir. 1986); Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1246-47, 1249 (3d Cir. 1983); Williams Elecs., Inc. v. Artic Int'l, Inc., 685 F.2d 870, 876-77 (3d Cir. 1982); Stern Elecs., Inc. v. Kaufman, 669 F.2d 852, 855 n.3 (2d Cir. 1982).


4. 49 F.3d 807 (1st Cir. 1995), aff'd by an equally divided Court, 516 U.S. 233 (1996).

289
answer is none.

In *Lotus*, the Court of Appeals for the First Circuit held that the menu hierarchy of the Lotus 1-2-3 spreadsheet program is not eligible for copyright protection because it is a "method of operation." The Supreme Court granted certiorari and affirmed the decision of the First Circuit by an equally divided court in a two sentence opinion without elucidation. Because the Supreme Court affirmed the First Circuit's holding in *Lotus* without any explanation, the current status of the law remains unclear. Without further clarification from the Supreme Court, only the most narrow, straightforward reading of the First Circuit's judgment is authoritative. Specifically, the *Lotus* court held only that the menu hierarchy in the Lotus 1-2-3 computer program is not subject to copyright protection because it is a method of operation. The First Circuit did not state that all nonliteral elements of computer programs fall into the categories excluded from protection by the Copyright Act. In fact, the First Circuit implied the opposite by indirectly acknowledging the Second Circuit's Abstraction-Filtration-Comparison ("A-F-C") test for determining copyright infringement of nonliteral elements of computer programs. The Second Circuit had formulated this test in *Computer Associates International, Inc. v. Altai, Inc.*, thereby indicating a belief that nonliteral elements of computer programs could be the subject of copyright protection.

Nonetheless, the First Circuit's reasoning in *Lotus* and the Supreme Court's minimalist affirmation of that case imply that the copying of nonliteral elements of computer programs might never constitute copyright infringement. This comment argues that such an interpretation of *Lotus* results in the appropriate legal treatment of computer programs. Part I of this comment provides a brief technical

5. *Id.* at 815.
8. *See Lotus*, 49 F.3d at 815.
9. *See id.*
11. *See Lotus*, 49 F.3d at 814-15 (considering the Second Circuit's test and holding that it is not applicable to the instant fact pattern).
12. 982 F.2d at 706-11.
13. The fact that the Second Circuit formulated a test to determine whether nonliteral elements of a computer program are copyright protectable and then applied that test to the instant fact pattern clearly indicates that the Second Circuit believed that nonliteral elements could be copyright protectable.
overview of computer programs. Part II presents a brief history of the relevant technology, examines the protection of nonliteral elements of literary works under copyright law, and outlines the copyright treatment of nonliteral elements of computer programs prior to *Lotus*. Part III outlines the procedural history of *Lotus* and details the holding of the Court of Appeals for the First Circuit in that case. Part IV examines the current status of the law. Lastly, Part V thoroughly analyzes the conclusions and reasoning of the circuit court’s opinion in *Lotus*. Part V demonstrates the appropriateness of the circuit court’s holding and looks at its implications for the copyright protection of nonliteral elements of computer programs.

I. TECHNOLOGICAL BACKGROUND

The Copyright Act defines a computer program as “a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.” Although not inaccurate, this definition does not explain enough to analyze the issues raised by *Lotus*. At its lowest level, a computer program is a collection of discrete voltage states that constitute a series of instructions that when executed will cause a computer to do something. Human beings conceive of these voltage states as zeros and ones to make this process more readily understandable. The zeros and ones are in reality metaphors for the underlying low voltages and high voltages. A series of zeros and ones in a specific order constitutes an instruction to the computer. Programmers entered the very first computer programs as strings of ones and zeros, or binary. Because this process was extremely labor intensive, programmers developed tools to allow the writing of computer programs in higher level computer languages that more closely resemble human language. Using terms such as “if,” “else,” “begin,” “end,” and assorted mathematical and logical operators, computer programmers are more easily able to create large programs. The text that represents the computer program in this form is called “source


15. For example, on an Intel 8086 microprocessor, the binary sequence “10011001” tells the computer to execute a CWD, or Convert Word to Double, instruction. See STEPHEN P. MORSE, THE 8086/8088 PRIMER: AN INTRODUCTION TO THEIR ARCHITECTURE, SYSTEM DESIGN, AND PROGRAMMING 54 (2d ed. 1982).

16. The numbering system which consists of only zeros and ones is called binary, or base two. Information on a computer is represented by a sequence of two distinct voltage states, which in turn are represented by zeros and ones. See id. at 3-4.
The source code is fed into a special computer program, called a compiler, that translates each line of source code into the actual instructions to the computer that it represents. This binary output from the compiler is called “object code.” Computers only run programs in binary form. Computer programmers can write source code in a high-level language such as C++ or Pascal, in which each line of source code represents multiple instructions to the computer, or in assembly language, in which, generally, each line of source code represents one actual instruction to the computer.

II. HISTORICAL BACKGROUND

A. Brief History of Relevant Technology

Today, corporations utilize computers heavily, and the sale of commercial software is an enormous business. Consequently, the legislative and judicial classification and treatment of computer programs is an issue of enormous legal and economic importance. Yet, computer use became very widespread in a relatively short period of time. In the 1950s, state of the art computers cost millions of dollars and took up multiple rooms. With the introduction of the integrated circuit in 1959 and the subsequent development of the microprocessor—which was first commercially utilized in 1970 by the now famous Intel corporation—computers quickly became smaller, cheaper, and much more powerful.

17. I have intentionally avoided an explanation of linking and nonexecutable object modules versus executable images. Such distinctions are important generally, but are outside the scope of this comment.

18. The first viable electronic computer, the ENIAC, was completed in 1945. See Jon Palfreman & Dorn Swade, The Dream Machine: Exploring the Computer Age 8, 44 (1991). It was one hundred feet long, eight feet high, three feet deep, weighed thirty tons, and contained 18,000 vacuum tubes. See id. at 44. It cost 500,000 1945 dollars. See id.

19. See Morse, supra note 15, at 8.

20. An integrated circuit (“IC,” or chip) is the combination of many separate electronic components, such as transistors, resistors, and capacitors, into a single complex electronic circuit, fixed on a wafer of silicon. See id.

21. See Palfreman & Swade, supra note 18, at 91.

22. A microprocessor is a single integrated circuit that contains all of the components necessary to constitute the control unit of a general purpose programmable computer. A microprocessor is sometimes called “a computer on a chip.” See Morse, supra note 15, at 8; Palfreman & Swade, supra note 18, at 107.

23. See Palfreman & Swade, supra note 18, at 107.

24. The Altair, the first computer built around a microprocessor and available for personal home use, was offered for sale in 1975. See id. at 109. It cost 500 1975 dollars. See id. It was advertised for sale in Popular Mechanics magazine, essentially as a toy for hobbyists. See id. Today, a computer with hundreds or even thousands of times more computing power than the computers that controlled the Apollo space flights to the moon can be purchased for a thousand
Technology changes more quickly than law. The legal system addresses new technologies by attempting to analogize them to things for which legal rules already exist. The Copyright Act and its associated legislative history classify computer programs as literary works and, thus, extend copyright protection to computer programs. This legal classification, while not necessarily inappropriate, requires courts to perform some new analysis since computer programs are in various ways different from traditional literary works.

B. Copyright Protection for Nonliteral Elements of Literary Works

Copyright infringement of literary works is not limited to a copying of the words but can also be achieved by a copying of nonliteral elements, such as plots or characters. Courts have long held that the copying of nonliteral elements of literary works can constitute copyright infringement. For example, in Nichols v. Universal Pictures Corp., Judge Learned Hand stated that “[copyright protection] cannot be limited literally to the text, else a plagiarist would escape by immaterial variations.” Nichols concerned nonliteral similarities between two plays. In Twentieth Century-Fox Film Corp. v. MCA, Inc., the Ninth Circuit found that plot similarities between Battlestar Galactica and Star Wars could constitute the basis for a finding of copyright infringement. In Sid & Marty Krofft Television Productions, Inc. v. McDonald’s Corp., the Ninth Circuit held copyright infringing similarities between the McDonaldland characters and the characters from the H.R. Pufnstuf television program were established by the “‘total concept and feel’” of the productions.

In Nichols, Judge Hand acknowledged that determining where the copying of nonliteral elements constitutes copyright infringement is an amorphous process. As Judge Hand eloquently phrased the issue, “as soon as literal appropriation ceases to be the test, the whole

dollars.

25. See supra note 1 and accompanying text.
26. 45 F.2d 119 (2d Cir. 1930).
27. Id. at 121.
28. See id. at 120.
29. 715 F.2d 1327 (9th Cir. 1983).
30. See id. at 1329.
31. 562 F.2d 1157 (9th Cir. 1977).
32. See id. at 1167 (quoting Roth Greeting Cards v. United Card Co., 429 F.2d 1106, 1110 (9th Cir. 1970)).
33. See Nichols, 45 F.2d at 121.
matter is necessarily at large." The difficulty is manifested in attempting to distinguish between protectable expression and a nonprotectable idea. Judge Hand enunciated the following test for attempting to make this determination:

Upon any work, and especially upon a play, a great number of patterns of increasing generality will fit equally well, as more and more of the incident is left out. The last may perhaps be no more than the most general statement of what the play is about, and at times might consist only of its title; but there is a point in this series of abstractions where they are no longer protected, since otherwise the playwright could prevent the use of his "ideas," to which, apart from their expression, his property is never extended.

This abstraction test attempts to isolate protectable expression that exists in the gray area between the literal elements of a work and the nonprotectable ideas of the work. Judge Hand instructed that the line beyond which protection does not exist is not a bright one. Referring to the point at which expression merges with idea, Judge Hand stated, "Nobody has ever been able to fix that boundary, and nobody ever can."

C. Copyright Protection for Nonliteral Elements of Computer Programs Prior to Lotus

The classification of computer programs as literary works has resulted in some confusion as to where copyright protection is appropriate for the nonliteral elements of computer programs. Computer programs are legally classified as literary works and, as such, are the subject of copyright protection. The literal elements of a computer program consist of the source code and the object code. The copying of the code of a computer program is analogous to the copying of the words of a traditional literary work. However, as courts have extended copyright protection to the nonliteral elements of traditional literary works, courts have extended copyright protection to the nonliteral elements of computer programs as well.

In 1986 in Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc., the Third Circuit performed the first thorough copyright infringement analysis of the copying of nonliteral elements of

---

34. Id.
35. Id.
36. Id.
38. 797 F.2d 1222 (3rd Cir. 1986).
computer programs. In the facts of Whelan, the Jaslow Dental Laboratory hired the Strohl Systems group to write a computer program to automate certain business functions of the laboratory. The agreement between the parties stated that Strohl would retain ownership of the software it developed for Jaslow. Pursuant to this agreement, Strohl officer and half owner Elaine Whelan developed the Dentalab computer program for use by Jaslow. Whelan left Strohl and formed Whelan Associates, which acquired Strohl's interest in the Dentalab program. While using the Dentalab software, Jaslow proceeded to write the Dentcomm computer program. Dentcomm was functionally similar to Dentalab, but Jaslow wrote it in a different programming language with the intention of making it usable on a wider variety of computers. Jaslow terminated its business relationship with Whelan and began marketing the Dentcomm software. Consequently, Whelan brought an action alleging that the Dentcomm computer program infringed Whelan's copyright in Dentalab.

The district court found that Jaslow did not copy the Dentalab source code and that Jaslow even wrote the allegedly infringing computer program in a different programming language. Nonetheless, the district court held that the structure and organization of the two computer programs were substantially similar. The district court found that Dentcomm infringed Whelan's copyright in Dentalab because the two programs were substantially similar and because Jaslow had access to Whelan's computer program. To determine that copyright infringement had occurred, the district court used a traditional "access plus substantial similarity" test, but with expert testimony to establish the existence of the substantial similarity.

39. See id. at 1225.
40. See id. at 1225 n.2.
41. See id. at 1225-26.
42. See id. at 1226.
43. See id.
44. Strohl Systems wrote Dentalab in EDL (Event Driven Language). See id. at 1226. Jaslow wrote Dentcomm in BASIC. See id.
45. See id.
46. See id.
47. See id. at 1226-27.
48. See id. at 1227.
50. See id. at 1322.
51. See id. at 1321-22.
52. See id. at 1316, 1321-22.
On appeal, the Court of Appeals for the Third Circuit affirmed the district court's finding of copyright infringement. The court explicitly held that copyright protection for computer programs extends to the nonliteral structural and organizational elements of the program and is not limited to a program's source or object code. In its affirmation of the district court, the circuit court established a methodology for distinguishing protectable expression from nonprotectable ideas for utilitarian works such as computer programs. Drawing on Baker v. Selden, the court held that the purpose of a utilitarian work is the idea of the work. Thus, it is appropriate to articulate the purpose of the work, and then distinguish between the elements that are necessary for the purpose and those that are not. The former elements are necessary for the idea of the work and, thus, merge with the idea of the work. Therefore, they are not copyrightable. The latter elements are not part of the idea but instead are expressions of the idea and, as such, are the subject of copyright protection. Under this reasoning, nonliteral elements of computer programs are protectable where they are not needed to achieve the program's desired result, but instead are either superfluous or are among several possible ways of achieving that desired result.

Initially, several district courts not under the jurisdiction of the Third Circuit followed Whelan. Then in 1992, in Computer Associates International, Inc. v. Altai, Inc., the Court of Appeals for the Second Circuit rejected the Whelan test. The Second Circuit held that Whelan falsely assumed that every computer program had only one idea and that the Whelan court, through its misuse of terms of art, had demonstrated a "flawed understanding of a computer program's
In the facts of Altai, Computer Associates wrote a computer program called ADAPTER. Computer Associates wrote ADAPTER as a subprogram of a job scheduling program for IBM mainframe computers called CA-SCHEDULER. The CA-SCHEDULER program created and implemented a schedule dictating when the computer should perform various operations. The ADAPTER subprogram functioned as a translator program that allowed CA-SCHEDULER to run on a variety of different computer systems by translating the general directives of CA-SCHEDULER into the specific instructions necessary to interact with the various systems.

Altai had developed its own job scheduling program called ZEKE. In order to develop its own adapter type translator program for ZEKE, Altai hired Claude F. Arney, an employee of Computer Associates. Arney had worked on ADAPTER for Computer Associates. When Arney left Computer Associates to work for Altai, he took copies of the ADAPTER source code. Arney wrote a translator program called OSCAR 3.4 for Altai. Without the knowledge of anyone else at Altai, Arney copied portions of the ADAPTER source code to create OSCAR 3.4. Arney copied approximately thirty percent of the source code of OSCAR 3.4 from ADAPTER. Computer Associates learned of this copying and brought a copyright infringement action against Altai.

Upon learning of Arney's copying of the ADAPTER source code, management at Altai initiated a rewrite of OSCAR. Management removed all portions of OSCAR that had been copied from the ADAPTER source code. Altai brought in a team of eight computer programmers who had not previously been involved in the project to

65. Id. at 705-06.
66. See id. at 698.
67. See id.
68. See id.
69. See id. at 698-99.
70. See id. at 699.
71. See id.
72. See id.
73. See id. at 699-700.
74. See id. at 700.
75. See id.
76. See id.
77. See id.
78. See id.
79. See id.
reconstruct the excised portions of OSCAR 3.4.\textsuperscript{80} Over a six month period, this team replaced all of the sections of OSCAR 3.4 that Arney had copied from the ADAPTER source code.\textsuperscript{81} They did not have access to the ADAPTER source code, but instead wrote original source code to mimic the functionality of ADAPTER.\textsuperscript{82} The resulting program was called OSCAR 3.5.\textsuperscript{83} Altai took OSCAR 3.4 off the market and sent copies of OSCAR 3.5 to all users of OSCAR 3.4.\textsuperscript{84}

Computer Associates alleged that both the copying of the ADAPTER source code in OSCAR 3.4 and the copying of the nonliteral elements of the ADAPTER program in OSCAR 3.5 infringed its copyright in ADAPTER.\textsuperscript{85} The district court found that the copying of the ADAPTER source code constituted infringement, and Altai did not contest this finding.\textsuperscript{86} On the matter of the copying of nonliteral elements, the district court found that no infringement had occurred, and Computer Associates appealed.\textsuperscript{87}

The Court of Appeals for the Second Circuit affirmed the holdings of the district court.\textsuperscript{88} In so doing, the circuit court established the A-F-C test for determining copyright infringement of nonliteral elements of computer programs.\textsuperscript{89} The A-F-C test is instructed by Judge Hand’s abstraction analysis from Nichols.\textsuperscript{90} This three-part test first involves abstracting backwards in stages from the literal instructions of the allegedly infringing computer program, to the components or subroutines of the program, to the design documents and flowcharts describing the program, and eventually to the function or purpose of the program, separated from the program itself.\textsuperscript{91} Once the abstraction to the program’s function has been performed, the second step is to filter out any ideas, utility merged into ideas, and other noncopyrightable material, such as that which was taken from the public domain.\textsuperscript{92} The third step is to compare whatever protectable expression survived the filtration with the

\textsuperscript{80} See id.
\textsuperscript{81} See id.
\textsuperscript{82} See id.
\textsuperscript{83} See id.
\textsuperscript{84} See id.
\textsuperscript{85} See id. at 700-01.
\textsuperscript{86} See id. at 701.
\textsuperscript{87} See id.
\textsuperscript{88} See id. at 697.
\textsuperscript{89} See id. at 706-11.
\textsuperscript{90} See id. at 706-07.
\textsuperscript{91} See id.
\textsuperscript{92} See id. at 707-10.
allegedly infringed work for substantial similarity. Although this standard clearly allows for the possibility of copyright protection for nonliteral elements of computer programs, the court recognized that the level of protection that exists might be minimal and that the issue is a vague one requiring further development over time. The court stated:

To be frank, the exact contours of copyright protection for nonliteral program structure are not completely clear. We trust that as future cases are decided, those limits will become better defined. Indeed, it may well be that the Copyright Act serves as a relatively weak barrier against public access to the theoretical interstices behind a program's source and object codes.

Interestingly, the Second Circuit found that the copying of nonliteral elements of the computer program in the instant case did not constitute copyright infringement.

The Second Circuit's holding in *Altai* has been extremely influential in the other circuits. Since *Altai*, every circuit that has explicitly considered the question of copyright protection of nonliteral elements of computer programs has endorsed the A-F-C test, with the exception of the First Circuit in *Lotus*. The Federal Circuit, the Fifth Circuit, the Ninth Circuit, the Tenth Circuit, and the Eleventh Circuit endorsed the *Altai* test. The Third Circuit continues to maintain its test from *Whelan*. The most substantial and interesting move away from the A-F-C test as a means of determining copyright infringement of nonliteral elements of computer programs came in the First Circuit's holding in *Lotus*, which the next Part of this comment considers in detail.

93. See id. at 710-11.
94. Id. at 712.
95. See id. at 715.
96. See Atari Games Corp. v. Nintendo of America, 975 F.2d 832, 839 (Fed. Cir. 1992).
100. See Mitek Holdings, Inc. v. Arce Eng'g, Co., 89 F.3d 1548, 1555-56 (11th Cir. 1996); Bateman v. Mnemonics, Inc., 79 F.3d 1532, 1543-45 (11th Cir. 1996).
III. THE LOTUS CASE

A. Facts

Lotus 1-2-3 is a computer spreadsheet program that allows users to execute various accounting functions automatically on a computer.\(^\text{102}\) Users operate the program through a series of menu commands associated with certain functions, such as "Copy," "Print," or "Quit."\(^\text{103}\) Users execute commands by selecting them on the screen or by typing their first letter.\(^\text{104}\) Lotus 1-2-3 has a total of 469 commands, arranged hierarchically into more than fifty menus and submenus.\(^\text{105}\)

Lotus 1-2-3 allows users to write macros, which are simple programs consisting of a series of commands. The user describes a consecutive series of commands and saves the information in a file. The user may then run these commands by typing a single keystroke, rather than executing each command individually. Lotus 1-2-3 includes the ability to read and execute such macros.\(^\text{106}\)

In 1987, Borland International released Quattro, a different and competing spreadsheet program.\(^\text{107}\) Quattro was developed by programmers at Borland without access to the 1-2-3 source code. Quattro includes a great deal of functionality not present in 1-2-3.\(^\text{108}\) Additionally, Quattro includes "a virtually identical copy of the entire 1-2-3 menu tree,"\(^\text{109}\) as well as the ability to read and execute Lotus macros. Borland did not copy any of the Lotus source code, but instead wrote its own program to mimic the Lotus menu.\(^\text{110}\) This was done so that users already familiar with the Lotus menu could use Quattro without having to learn a new menu system.\(^\text{111}\) Quattro also provided a completely different menu of its own.\(^\text{112}\)

\(^{103}\) See id.
\(^{104}\) See id.
\(^{105}\) See id.
\(^{106}\) See id.
\(^{107}\) See id.
\(^{108}\) See id.
\(^{109}\) See id. at 810.
\(^{110}\) See id.
\(^{111}\) See id.
\(^{112}\) See id.
B. Procedural History

On June 28, 1990, in the separate case of Lotus Development Corp. v. Paperback Software International,\textsuperscript{113} the United States District Court for the District of Massachusetts held that the Lotus 1-2-3 menu structure was expression protected by copyright.\textsuperscript{114} The next day, Borland filed a declaratory judgment action against Lotus in the Northern District of California, seeking a declaration of noninfringement.\textsuperscript{115} Three days later, on July 2, 1990, Lotus brought an action against Borland in the District of Massachusetts for infringement of copyright.\textsuperscript{116} On September 10, 1990, the district court in California dismissed Borland's declaratory judgment in favor of the infringement action.\textsuperscript{117}

Lotus and Borland both moved for summary judgment, and both motions were denied on March 20, 1992.\textsuperscript{118} However, the district court invited the parties to file renewed "better focus[ed]" summary judgment motions.\textsuperscript{119} On April 24, 1992, both parties filed new summary judgment motions.\textsuperscript{120} Borland claimed that the Lotus 1-2-3 menus were not copyrightable as a matter of law.\textsuperscript{121} Lotus claimed that Borland had copied the entire 1-2-3 user interface, thereby infringing Lotus' copyrights.\textsuperscript{122}

On July 31, 1992, the district court denied Borland's motion for summary judgment and granted Lotus' summary judgment motion in part.\textsuperscript{123} The court reasoned that the 1-2-3 menu hierarchy is copyrightable expression because various other menu hierarchies are possible for spreadsheet programs.\textsuperscript{124} The court, however, held that a reasonable jury could conclude that Borland had not copied the entirety of the 1-2-3 menu, as Lotus had claimed.\textsuperscript{125} The court concluded that Borland had infringed Lotus' copyright based on the extent to which Borland had copied expressive elements of the 1-2-3

\textsuperscript{114} See id. at 68.
\textsuperscript{115} See Lotus, 49 F.3d at 807.
\textsuperscript{116} See id.
\textsuperscript{117} See id.
\textsuperscript{119} See id. at 82.
\textsuperscript{120} See Lotus, 49 F.3d at 810.
\textsuperscript{121} See id.
\textsuperscript{122} See id.
\textsuperscript{124} See id. at 217.
\textsuperscript{125} See id. at 209.
menu structure that are integral parts of Lotus 1-2-3.\textsuperscript{126} The court further concluded that a jury trial was necessary to determine the amount of protected expression that Borland had actually copied.\textsuperscript{127}

Following the district court’s grant of partial summary judgment for Lotus, Borland removed the Lotus emulation menu from its Quattro program.\textsuperscript{128} Borland did, however, retain in Quattro the capability to run Lotus macros.\textsuperscript{129} Consequently, the district court permitted Lotus to file an additional complaint claiming that Borland’s inclusion of the macro running functionality in Quattro infringed the Lotus 1-2-3 copyright.\textsuperscript{130} The parties agreed to try the outstanding issues without a jury.\textsuperscript{131} The district court held two trials. The first trial considered the extent to which Borland copied the Lotus 1-2-3 menu.\textsuperscript{132} The second trial addressed the issues of the supplemental complaint concerning infringement by Borland’s Lotus 1-2-3 macro reader, which Borland called “Key Reader.”\textsuperscript{133}

In the first trial, the district court held that “each of the Borland emulation interfaces contains a virtually identical copy of the 1-2-3 menu tree and that the 1-2-3 menu tree is capable of a wide variety of expression.”\textsuperscript{134} In the second trial, the district court held that Borland’s Key Reader infringes Lotus’ copyright.\textsuperscript{135} The court held that the Key Reader included the Lotus 1-2-3 menu tree structure,\textsuperscript{136} which is a protectable element of Lotus 1-2-3.\textsuperscript{137} The district court issued a permanent injunction against Borland.\textsuperscript{138} Borland proceeded to appeal. The issue considered on appeal was Borland’s copying of the Lotus 1-2-3 menu command hierarchy.\textsuperscript{139}

\begin{footnotes}
\footnotetext[126]{See id. at 223.}
\footnotetext[127]{See id.}
\footnotetext[128]{See Lotus Dev. Corp. v. Borland Int’l, Inc., 49 F.3d 807, 811 (1st Cir. 1995), aff’d by an equally divided Court, 516 U.S. 233 (1996).}
\footnotetext[129]{See id. at 811-12.}
\footnotetext[130]{See id. at 812.}
\footnotetext[131]{See id.}
\footnotetext[132]{See id.}
\footnotetext[133]{See id.}
\footnotetext[135]{See id. at 228.}
\footnotetext[136]{See id. at 233.}
\footnotetext[137]{See id. at 245.}
\footnotetext[138]{See id. at 233.}
\footnotetext[139]{See Lotus Dev. Corp. v. Borland Int’l, Inc., 49 F.3d 807, 812 (1st Cir. 1995), aff’d by an equally divided Court, 516 U.S. 233 (1996).}
\end{footnotes}
C. The Holding of the Court of Appeals for the First Circuit

On appeal, Borland did not dispute that Lotus held a valid copyright on the Lotus 1-2-3 program as a whole.\textsuperscript{140} Borland conceded further that it had copied the menu command hierarchy of Lotus 1-2-3.\textsuperscript{141} In its defense, Borland argued that it had not infringed Lotus’ copyright because the menu command hierarchy is not protectable because it is a system, method of operation, process, or procedure,\textsuperscript{142} all of which are excluded from copyright protection by the Copyright Act.\textsuperscript{143} The First Circuit stated that the question before the court was whether “a menu command hierarchy standing on its own . . . without other elements of the user interface, such as screen displays” constitutes copyrightable subject matter.\textsuperscript{144}

The First Circuit considered the applicability of the Second Circuit’s A-F-C test.\textsuperscript{145} The court acknowledged that the Second Circuit’s test might be appropriate for determining the nonliteral copying of a computer program’s code but held that it was not applicable to the instant case.\textsuperscript{146} The court reasoned that Borland’s copying from Lotus was not “nonliteral copying of computer code,” but “literal copying of . . . [a] menu command hierarchy.”\textsuperscript{147}

The First Circuit held that the Lotus menu command hierarchy is a nonprotectable method of operation.\textsuperscript{148} The court found that the menu command hierarchy is the method by which the program is controlled.\textsuperscript{149} The concurring judge analogized the menu command hierarchy to the buttons on a video cassette recorder\textsuperscript{150} and to the layout of the keys on a typewriter.\textsuperscript{151}

The court distinguished between the menu hierarchy itself, through which the user controls the software, and the screen displays, which the user views but does not actually use to manipulate the program.\textsuperscript{152} The court rejected the district court’s assertion that the

\textsuperscript{140} See id. at 813.
\textsuperscript{141} See id.
\textsuperscript{142} See id. at 812.
\textsuperscript{143} See 17 U.S.C. § 102(b) (1994).
\textsuperscript{144} See Lotus, 49 F.3d at 813.
\textsuperscript{145} See id. at 814-15.
\textsuperscript{146} See id. at 815.
\textsuperscript{147} Id. at 814.
\textsuperscript{148} See id. at 815.
\textsuperscript{149} See id.
\textsuperscript{150} See id. at 817 (Boudin, J., concurring).
\textsuperscript{151} See id. at 819-20 (Boudin, J., concurring).
\textsuperscript{152} See id. at 815-16.
specific arrangement of the menu command hierarchy constitutes a protectable expression of an idea. The court held that the menu command hierarchy is the means by which a program is operated and, as such, is a nonprotectable method of operation, regardless of the fact that it could have been designed differently and that other arrangements for menu command hierarchies exist beside this one.

D. The Supreme Court’s Sparse Opinion

The Supreme Court granted certiorari and proceeded to affirm the decision of the First Circuit in a two sentence opinion stating only: "The Judgment of the United States Court of Appeals for the First Circuit is affirmed by an equally divided Court. Justice Stevens took no part in the consideration or decision of this case.'

IV. CURRENT STATUS OF THE LAW

The holding in Lotus can be interpreted to indicate that the copying of nonliteral elements of computer programs can never constitute copyright infringement. However, despite the Supreme Court’s affirmation of Lotus, three circuits have subsequently endorsed the A-F-C test for determining copyright infringement for the copying of nonliteral elements of computer programs.

In Softel, Inc. v. Dragon Medical & Scientific Communications, Inc., the Second Circuit held that “[i]t is well established in this Circuit that nonliteral similarity of computer programs can constitute copyright infringement." The Second Circuit proceeded to cite the A-F-C test established in Altai as the proper means for determining such infringement. Thus, the Second Circuit, as the originator of the A-F-C test, was not persuaded by the analysis of the First Circuit in Lotus.

In Mitel, Inc. v. Iqtel, Inc., the Tenth Circuit explicitly declined to follow Lotus and held that despite the wording of the Copyright Act, elements of a method of operation may be subject to copyright

153. See id. at 816.
154. See id.
157. 118 F.3d 955 (2d Cir. 1997).
158. Id. at 963.
159. See id.
160. 124 F.3d 1366 (10th Cir. 1997).
protection. Interestingly, the Tenth Circuit found no elements so protectable in the case at bar.

In *Bateman v. Mnemonics, Inc.*, the Eleventh Circuit endorsed the A-F-C test. More recently, in *Mitek Holdings, Inc. v. Arce Engineering Co.*, the Eleventh Circuit found that a specific computer program's menu structure was a process and, thus, not eligible for copyright protection. The *Mitek* court did not go so far as to hold that software menus are noncopyrightable processes as a matter of law.

V. ANALYSIS

A. Lotus as a Rejection of Copyright Protection for the Copying of Nonliteral Elements of Computer Programs

Despite the continued use of the A-F-C test by some circuits, *Lotus* raises the interesting question of whether the copying of nonliteral elements of computer programs can ever constitute copyright infringement. Although the *Lotus* court does not explicitly state that it cannot, its analysis logically suggests that with an exception for artistic screen images, which are protectable as audiovisual works, copying of nonliteral elements of computer programs never infringes current copyright laws. The *Lotus* court classified the copying at bar as the "literal copying of ... [a] menu command hierarchy" as distinguished from "the nonliteral copying of computer code." It is important, therefore, to examine the exact nature of the copying in *Lotus*.

It is not contested that Borland did not copy the underlying code of the Lotus 1-2-3 computer program. Such copying would constitute infringement and would be actionable. Instead, Borland wrote its own computer program which mimicked both specific visual

---

161. See id. at 1372.
162. See id. at 1376.
163. 79 F.3d 1532 (11th Cir. 1996).
164. See id. at 1543-45.
165. 89 F.3d 1548 (11th Cir. 1996).
166. See id. at 1557.
167. See id.
168. See supra text accompanying notes 140-54.
170. See id. at 816.
171. See supra note 2 and accompanying text.
and nonvisual components of the Lotus program. Borland simulated both the appearance and behavior of the Lotus menu system and the Lotus macro reader. Without access to the source code for Lotus 1-2-3, Borland wrote a computer program to do exactly what parts of the Lotus program do. The parts of Lotus 1-2-3 simulated by Borland, the menu hierarchy and the macro reader, constitute the way in which a user operates Lotus 1-2-3. Thus, the court chose to classify the writing of a work-a-like menu and macro reader as a literal copying of that method of operation.\footnote{See Lotus, 49 F.3d at 814.} This is not inaccurate, but it is not the whole story.

The terms "literal copying" and "the copying of literal elements" are often used interchangeably but in fact have different meanings. Likewise, the terms "nonliteral copying" and "the copying of nonliteral elements" are often mistakenly used interchangeably. In the instant case, the source code that generates the Lotus 1-2-3 menu hierarchy is a literal element of a computer program. A literal copying of each line of that code would constitute copyright infringement. Less obviously but equally accurately, a nonliteral copying of the source code (a literal element) could also constitute copyright infringement. If a plagiarist with access to the source code made a copy with intentional minor variations, the result would constitute a nonliteral copy of a literal element. Such nonliteral copying of source code is analogous to the copying of text with the substitution of synonyms for some of the original words.

The copying of nonliteral elements can also be literal or nonliteral. The idea of the Lotus 1-2-3 menu hierarchy is a nonliteral element of the Lotus 1-2-3 program, distinguishable from the expression of that idea fixed in source code. A work-a-like program is a literal copying of the underlying idea (a nonliteral element) of the original program. Nonliteral elements could also be copied nonliterally, by the writing of a program that imitates the original program less exactly. The \textit{Lotus} appellate court was entirely correct in stating that the copying at bar was the literal copying of a method of operation, and not the nonliteral copying of code. The copying in \textit{Lotus} was also not the literal copying of computer code. The copying was not literal or nonliteral copying of literal elements of Lotus 1-2-3 but instead was the literal copying of nonliteral elements.

It is of legal significance that the court held that the nonliteral elements that were copied constituted a nonprotectable method of
operation. It is well established that although original expressions of ideas are protectable by copyright, there is no copyright protection for ideas themselves.173 The fact that there is no copyright protection for "any idea," as well as for any "procedure, process, system, method of operation, concept, principle, or discovery" is also articulated in section 102(b) of the Copyright Act.174 The legislative history of the Copyright Act indicates that the exceptions to copyright protection listed in section 102(b) are a legislative codification of the idea-expression dichotomy associated with *Baker v. Selden*.175 Thus, it is entirely accurate and noncontroversial to state, as the *Lotus* court did, that the copying of an idea (or any of the extensions of an idea articulated in section 102(b) of Copyright Act) does not constitute copyright infringement. I believe further that the copying of nonliteral elements of computer programs will always be reducible to the copying of ideas or to one of the section 102(b) idea-related categories.

To defend this thesis, it is necessary to examine carefully the nature of the elements that were copied in *Lotus*. It is easy to confuse the screen images that a computer program generates with the nonvisual components of the program. Borland wrote code to simulate both visual and nonvisual elements of the Lotus 1-2-3 menu hierarchy. The court in *Lotus* articulated that the issue at bar was not the copying of the screen displays but the copying of the underlying menu command hierarchy, independent of any visual screen representation thereof.176 The court specifically declined to take a position on whether the screen images could be the subject matter of copyright protection.177 Subsection D of this Part analyzes screen displays generated by computer programs in detail. At present, it is important to note that both the visual and nonvisual aspects of the Lotus 1-2-3 command hierarchy are nonliteral elements of the Lotus 1-2-3 program, distinguishable from the source code that generates them. The First Circuit considered only the copying of the nonvisual elements of the menu hierarchy.

What Borland copied was not the actual instructions to the computer that result in Lotus' manifestation of the 1-2-3 menu hierarchy, but instead the structure behind parts of the Lotus

176. See *Lotus*, 49 F.3d at 815-16.
177. See id. at 816 n.10.
program—the hierarchy and functionality that constitute the Lotus 1-2-3 menu system and macro command reader. The court of appeals correctly held that this structure is reducible to a method of operation. Computer programs are utilitarian works created not as artistic expression (although certainly the craft of computer programming is not without creativity) but solely in order to perform some function or functions. Thus, it seems that every part of a computer program except the instructions is an idea, method of operation, system, or process. Unlike traditional literary works, nonliteral elements of computer programs cannot be isolated from the literal elements in any form more tangible than that of idea. As Judge Hand articulated in Nichols, a traditional literary work can contain a copyright protectable gray area somewhere between the literal elements and the ideas. Computer programs have no such gray area. The only nonliteral elements that can be extracted from computer programs are ideas. The elements of authorship that infuse computer programs with the creative originality required for copyright protection as literary works are fleeting and can be fixed only as code.

Just as Borland did not infringe Lotus’ copyright by writing a program that mimicked parts of Lotus 1-2-3, Borland arguably would not have infringed Lotus’ copyright if it had written a program that mimicked every functional aspect of Lotus 1-2-3 but without copying any of the underlying code. Perhaps if the Supreme Court had been more verbose in its affirmation of Lotus, this would be clearly established as a matter of law today.

B. Lotus as a Rejection of the Whelan Test

Conceivably, this comment’s reading of Lotus may be too broad, and under certain circumstances the copying of nonliteral elements of computer programs should still constitute infringement. At the very least, however, the Lotus court clearly rejected the already widely criticized Whelan test by holding that the existence of multiple ways of performing a function does not allow copyright protection for that which is otherwise nonprotectable. The Whelan court seems to have misapplied the merger doctrine associated with the idea-expression dichotomy.

178. See Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930).
179. See Lotus, 49 F.3d at 816.
The merger doctrine provides that where only a limited number of ways of expressing an idea exist, the expression of that idea merges with the idea and, like the idea itself, is removed from the realm of copyright monopoly. However, this does not imply automatic copyright protection for anything that can be expressed in multiple ways. Numerous ways exist to express various facts, ideas, methods of operation, and the like, for which copyright protection simply does not exist regardless of the number of methods of expression possible. As the *Lotus* court pointed out, the fact that Borland could have labeled Quattro's copy functionality "Clone, Ditto, Duplicate, Imitate, Mimic, Replicate, [or] Reproduce" instead of "Copy" in no way "magically change[s] the noncopyrightable menu command hierarchy into copyrightable subject matter."

The Supreme Court's affirmation of *Lotus* should be read as the pounding of another nail into the *Whelan* test's coffin. It would have been desirable for the Supreme Court to have taken the opportunity to overrule the Third's Circuit's misapplication of the merger doctrine as a standard for determining software infringement.

C. The Abstraction-Filtration-Comparison Test after *Lotus*

The *Lotus* court indirectly acknowledged the Second Circuit's A-F-C test by stating that the test may be useful for determining nonliteral copying of a computer program's code, but found the test inapplicable to the instant case. The *Lotus* court held that not only would the A-F-C test not be helpful in determining whether the copying of a menu hierarchy constituted copyright infringement, but that the test might actually be misleading. Concerning the A-F-C test, the court stated:

>[I]n instructing courts to abstract the various levels, it seems to encourage them to find a base level that includes copyrightable subject matter .... We think that abstracting menu command hierarchies down to their individual word and menu levels and then filtering idea from expression at that stage, as . . . the Altai . . . test[ ] require[s], obscures the more fundamental question of whether a menu command hierarchy can be copyrightable at all. The initial

---

182. *Id.* at 816.
183. See *supra* notes 89-94 and accompanying text.
184. See *Lotus*, 49 F.3d at 815.
185. See *id*.
186. See *id*.
inquiry should not be whether individual components of a menu command hierarchy are expressive, but rather whether the menu command hierarchy as a whole can be copyrighted.187

Thus, the First Circuit's refusal to apply the A-F-C test to determine the copyrightability of nonliteral elements of computer programs might well be read as an indirect rejection of that test. The Lotus court chose not to abstract nonliteral elements to higher levels, but instead to focus on the copyrightability of the nonliteral elements themselves. The court held that the process of abstracting nonliteral elements may be misleading because it falsely implies that protectable expression necessarily exists within these elements. However, the First Circuit did not explicitly reject the A-F-C test for use under all circumstances.

With this background, it is worthwhile to examine the Second Circuit's test to consider if it is appropriate for determining the protectability of nonliteral elements of computer programs and, if so, whether any nonliteral element of any computer program could ever be protectable under that test. As explained previously, the A-F-C test involves abstracting from the most literal and basic elements of a computer program—the actual instructions that constitute the program—to the most general—the idea of a program to perform a certain function. It is helpful and appropriate to think of computer programs in these terms of abstraction, but once the abstraction and filtration have been performed, the comparison will always come up empty as far as protectable elements are concerned.

At its most basic level, a computer program is a logically contiguous series of discrete voltage states capable of causing a computer to do something. Abstracted backwards from that is the source code. At these levels, it is not contested that a copying of the expression constitutes infringement. Everything at a higher level of abstraction than the code seems reducible to an idea, process, or method of operation. Put simply, the idea of a computer program to perform some function or operation is always a nonprotectable idea. A specific implementation of that idea in source code is a protectable literary work. For example, Lotus' source code for the 1-2-3 menu hierarchy is protectable. The idea of the menu hierarchy, not only a menu hierarchy in general, but the Lotus menu hierarchy specifically, is not. If nonliteral components are never protectable after the filtration required by the A-F-C test, then the fact that Lotus did not
explicitly reject this test does not show that nonliteral elements of computer programs will or should ever be the subject of copyright protection.

The *Altai* court instructed future courts to dissect the structure of computer programs by performing a conceptual reverse engineering.\(^{188}\) The court discussed the identification of intermediate steps in program development, such as modules, parameter lists, design specifications, and flow-charts.\(^{189}\) Despite the court's instructions, everything the court listed is either part of the source code or an idea preceding any specific implementation of the program. It is not insignificant that the *Altai* court found no protectable elements as a result of performing the A-F-C test.\(^{190}\) Arguably every element of every computer program at a level of abstraction higher than the source code is a nonprotectable idea, procedure, process, system, or method of operation. The holding of the First Circuit in *Lotus* implied this. The grant of certiorari shows that the Supreme Court views this issue as important. It would have been beneficial if the Supreme Court had elucidated this matter. In order to clarify the confusion and disagreement that now exists in the circuits, the Supreme Court should grant certiorari to another software copyright case. If certiorari is granted, it would be appropriate for the Supreme Court to hold that copyright protection for computer programs is limited to protection against a copying of the code.

D. The Exception for Artistic Screen Images

As mentioned earlier, one element of computer programs other than the code that can be protectable by copyright is artistic screen images. Sufficiently artistic screen images generated by computer programs are copyright protectable as audio-visual works.\(^{191}\) Such screen images are not protectable as literal or nonliteral elements of computer programs, but instead as audio-visual works separate from the computer program that generated them.\(^{192}\)

It is important to distinguish between computer screen images as nonliteral elements of computer programs and sufficiently artistic,
computer-program-generated visual images as audio-visual works. Computer programs can contain both visual and nonvisual nonliteral elements. For example, the Lotus 1-2-3 menu hierarchy consists of not only the nonvisual underlying structure but also the visual manifestation of the menu hierarchy on a computer screen. Absent sufficient creative originality, screen images generated by computer programs are best classified solely as visual, nonliteral elements of those programs. Courts should not extend copyright protection to such images as components of literary works because they, like all nonliteral elements of computer programs, will always be reducible to ideas.

A computer screen image can also be conceived of as an audiovisual work, severable from the associated computer program. Thus, given the requisite level of creativity and originality, such images are the subject of copyright protection as audiovisual works, entirely separately from the computer program that generated them. They are subject to copyright protection for the same reasons as audio-visual works fixed in more traditional media such as oil paint on canvas.

The same standards of copyright protection should apply to digital visual works as those fixed in more familiar media. It is unclear exactly how much protection courts will allow over time, but the contents of a computer screen should not be copyright protectable per se. Instead, they should be subject to the same originality and creativity requirements as other audiovisual works. This form of copyright protection of digital works survives Lotus, but it is a separate issue from the copyright protection of computer programs.

E. The Appropriate Legal Treatment of Computer Programs

Some commentators have suggested that the legal system should develop an entirely new method of protection for computer programs, but this is neither realistic nor necessary. The European Union, Japan, and other members of the international community have already followed the American example of extending copyright protection to computer programs. Difficulties raised by the legal

193. See id.
194. See generally Apple Computer, Inc. v. Microsoft Corp., 35 F.3d 1435 (9th Cir. 1994).
196. See Jane C. Ginsburg, Four Reasons and a Paradox: The Manifest Superiority of Copyright over Sui Generis Protection of Computer Software, 94 COLUM. L. REV. 2559, 2559
protection of computer programs should be solved within the realm of copyright law if possible.\textsuperscript{197} The unique nature of computer programs certainly raises copyright complications. This comment's reading of \textit{Lotus} provides a solution.

The \textit{Whelan} test mistook nonprotectable ideas for protectable expression. The \textit{Altai} test attempted to remedy this problem by detailing a thoughtful process for filtering out all nonprotectable elements of computer programs and leaving only that which is protectable. \textit{Altai} failed to realize, however, that only the code is protectable. If the \textit{Altai} test is applied correctly, no protectable elements will survive the filtration step at a level of abstraction beyond the source code. The copyright minimalist interpretation of \textit{Lotus} outlined in this comment provides the solution, and it does so within the framework of existing copyright law. The object and source code of computer programs are literary works, and a copying thereof can constitute copyright infringement. Every element of a computer program at a higher level of abstraction than the code is "an idea, procedure, process, system, method of operation, [or] concept"\textsuperscript{198} and, thus, is excluded from copyright protection under section 102(b) of the Copyright Act.

\section*{Conclusion}

Computer programs are legally classified as literary works and, as such, are protected from a copying of their underlying code.\textsuperscript{199} The copying of nonliteral elements of traditional literary works can constitute infringement.\textsuperscript{200} Consequently, courts have extended copyright protection to the nonliteral elements of computer programs.\textsuperscript{201} Prior to \textit{Lotus}, courts dissected nonliteral elements of computer programs, searching for copyrightable material.\textsuperscript{202} In \textit{Lotus}, the First Circuit declined to engage in such analysis, and instead held that specific nonliteral elements of a computer program constituted a method of operation and, thus, were nonprotectable as a matter of law.\textsuperscript{203} The Supreme Court affirmed the First Circuit's opinion in

\begin{footnotesize}
\begin{enumerate}
\item The unique nature of computer programs certainly raises copyright complications. This comment’s reading of \textit{Lotus} provides a solution.
\item The \textit{Whelan} test mistook nonprotectable ideas for protectable expression. The \textit{Altai} test attempted to remedy this problem by detailing a thoughtful process for filtering out all nonprotectable elements of computer programs and leaving only that which is protectable. \textit{Altai} failed to realize, however, that only the code is protectable. If the \textit{Altai} test is applied correctly, no protectable elements will survive the filtration step at a level of abstraction beyond the source code. The copyright minimalist interpretation of \textit{Lotus} outlined in this comment provides the solution, and it does so within the framework of existing copyright law. The object and source code of computer programs are literary works, and a copying thereof can constitute copyright infringement. Every element of a computer program at a higher level of abstraction than the code is “an idea, procedure, process, system, method of operation, [or] concept” and, thus, is excluded from copyright protection under section 102(b) of the Copyright Act.
\item Computer programs are legally classified as literary works and, as such, are protected from a copying of their underlying code. The copying of nonliteral elements of traditional literary works can constitute infringement. Consequently, courts have extended copyright protection to the nonliteral elements of computer programs. Prior to \textit{Lotus}, courts dissected nonliteral elements of computer programs, searching for copyrightable material. In \textit{Lotus}, the First Circuit declined to engage in such analysis, and instead held that specific nonliteral elements of a computer program constituted a method of operation and, thus, were nonprotectable as a matter of law. The Supreme Court affirmed the First Circuit’s opinion in
\item \textit{Lotus} outlined in this comment provides the solution, and it does so within the framework of existing copyright law. The object and source code of computer programs are literary works, and a copying thereof can constitute copyright infringement. Every element of a computer program at a higher level of abstraction than the code is “an idea, procedure, process, system, method of operation, [or] concept” and, thus, is excluded from copyright protection under section 102(b) of the Copyright Act.
\item Computer programs are legally classified as literary works and, as such, are protected from a copying of their underlying code. The copying of nonliteral elements of traditional literary works can constitute infringement. Consequently, courts have extended copyright protection to the nonliteral elements of computer programs. Prior to \textit{Lotus}, courts dissected nonliteral elements of computer programs, searching for copyrightable material. In \textit{Lotus}, the First Circuit declined to engage in such analysis, and instead held that specific nonliteral elements of a computer program constituted a method of operation and, thus, were nonprotectable as a matter of law. The Supreme Court affirmed the First Circuit’s opinion in
\end{enumerate}
\end{footnotesize}
Arguably, because computer programs are utilitarian works, all of their nonliteral elements are reducible to ideas, procedures, processes, systems, methods of operation, and concepts. The Copyright Act exempts things so classified from protection. Unlike traditional literary works, computer programs have no gray area between the ideas and the literal elements. The literary qualities of computer programs are fleeting and can be fixed only as code. Every element of a computer program severed from the code is an idea or is in one of the Copyright Act section 102(b)'s idea-related categories.

Furthermore, the earlier tests that allowed copyright protection for the copying of nonliteral elements of computer programs are problematic. The Whelan test is a misapplication of the merger doctrine and has been widely criticized as such. The Altai test, if applied correctly, seems incapable of ever indicating actual infringement. Arguably, no copying of computer programs other than the copying of code should ever constitute copyright infringement. Despite Lotus, many of the circuits continue to apply the Altai test to determine copyright protection for nonliteral elements of computer programs. The Supreme Court should grant certiorari to another software copyright case and articulate a clear rule for the copyright protection of computer programs. An appropriate rule would be to limit copyright protection for computer programs to a copying of the code.

an equally divided Court, 516 U.S. 233 (1996).

204. See supra text accompanying notes 155-56.