No. 16-1808

UNITED STATES COURT OF APPEALS FOR THE FOURTH CIRCUIT

SAS INSTITUTE, INC.,
Plaintiff-Appellant,

- v. -

WORLD PROGRAMMING LIMITED,

Defendant-Appellee.

Appeal from the United States District Court for the
Eastern District of North Carolina at Raleigh
Case No. 5:10-cv-00025-FL

Brief of BSA | The Software Alliance As *Amicus Curiae* In Support Of Plaintiff-Appellant

Andrew J. Pincus
Paul W. Hughes
Jonathan Weinberg
MAYER BROWN LLP
1999 K Street, N.W.
Washington, D.C. 20006
(202) 263-3000

Counsel for *Amicus Curiae* BSA | The Software Alliance

CORPORATE DISCLOSURE STATEMENT

Pursuant to Rules 26.1 and 29(c) of the Federal Rules of Appellate Procedure, amicus states that BSA | The Software Alliance has no parent company. No publicly held company owns 10% or more of its stock.

TABLE OF CONTENTS

Page(s)

TABLE OF AUTHORITIES

Page(s)

Cases

Statutes, Rules and Regulations

Other Authorities

STATEMENT OF INTEREST

BSA | The Software Alliance ("BSA") is an association of the world's leading software and hardware technology companies.[1] BSA's members include Adobe, Ansys, Apple, Autodesk, Bentley Systems, CA Technologies, CNC/Mastercam, Datastax, Dell, IBM, Intuit, Microsoft, Minitab, Oracle, Salesforce, SAS Institute ("SAS"), Siemens Splunk, Symantec, Trend Micro, Trimble, and Workday.

On behalf of its members, BSA promotes policies that foster innovation, growth, and a competitive marketplace for commercial software and related technologies. BSA members rely on copyright protection to establish property rights in their critical assets and provide essential legal protection for their substantial investments in those assets. As a group, they hold a significant number of copyrights. Because copyright policy is vitally important to promoting the innovation that has kept the United States at the forefront of software development, BSA members have a strong stake in the proper functioning of the U.S. copyright system.

---

[1] This brief is accompanied by a motion requesting leave to file. No party's counsel authored this brief in whole or in part. No party, party's counsel, or any person other than *amicus* or its counsel contributed money intended to fund preparing or submitting this brief. *See* Fed. R. App. P. 29(c)(5).

In this case, the district court failed to undertake a proper analysis of the copyrightability of the software at issue. The decision thus creates legal uncertainty regarding the scope of copyright protections for works in software, which undermines incentives to develop new software and hampers innovation.

SUMMARY OF THE ARGUMENT

Just like any other original, creative expression, computer software is protected by copyright. Congress explicitly granted this copyright protection. And courts have repeatedly held that software copyrights extend not only to a program's "literal" elements—the source code and object code—but also to its nonliteral elements, such as structure, sequence, organization, user interface, screen displays, menu structures, and the like.

To determine which of a particular program's non-literal elements are copyrightable, a court must analyze the program to distinguish unprotectable ideas from protectable expression. Because this fact-specific analysis is not readily susceptible to bright-line rules, courts have developed specialized frameworks, such as the abstraction-filtration-comparison test, for drawing the necessary distinctions. That approach—adopted by the Second, Fifth, Ninth, Tenth, and Eleventh Circuits—calls for analyzing software at various levels of abstraction to identify the designer's creative choices at each step and to filter out non-copyrightable ideas. The Federal Circuit recently invoked that same approach in *Oracle v. Google* to determine that certain portions of an Application Programming Interface ("API") are protected by copyright. *Oracle Am., Inc. v. Google Inc.*, 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied sub nom. Google, Inc. v. Oracle Am., Inc.*, 135 S. Ct. 2887 (2015).

The district court's analysis in this case was insufficient because it did not attempt to distinguish protectable expression from unprotectable ideas in SAS's system. SAS accused World Programming Limited ("WPL") of infringing SAS's copyrights in, among other things, the collection of SAS Procedures (i.e., "PROCs") made available on its system. The district court held that those elements are non-copyrightable, but it did so without analyzing the PROCs to determine whether they include protectable expression. The court did not apply any accepted framework for accomplishing this separation, nor did it recognize that software can include both copyrightable and non-copyrightable expression. The district court also failed to meaningfully distinguish *Oracle*.

For these reasons, the Court should vacate the district court's grant of summary judgment, and it should remand this matter for an analysis in accordance with the prevailing legal standards.

ARGUMENT

I. Copyright Protection For Software Is Essential To Ensure Continued Innovation And Economic Growth.

A. The software industry is a key engine of economic growth.

Software is a critical driver of the modern economy in many different ways.

First, the software industry contributes more than a trillion dollars a year to the U.S. economy. BSA, *The $1 Trillion Economic Impact of Software*, at 3 (June 2016). That number includes $475.3 billion in direct contributions and over $525 million in indirect and induced contributions. *Id.*

The software industry makes these indirect contributions by providing the backbone for virtually all businesses, supporting critical business functions, including "finance, human resources, operations and logistic, sale and market." Santanu Kumar Misra & Amitava Ray, "Integrated AHP-TOPSIS Model for Software Selection Under Multi-criteria Perspective," *Driving the Economy Through Innovation and Entrepreneurship* 879, 879 (2013). From the agricultural sector (where software helps farmers increase crop yields) to the healthcare sector (where software drives diagnostic accuracy), and the public sector (where software helps governments deliver services, reduce traffic congestion, fight crime, and cut costs), software creates efficiencies across nearly every aspect of the nation's economy. BSA, *The $1 Trillion Economic Impact of Software*, at 7-9.

Software also helps companies "collaborate more effectively internally and externally, scale operations faster, operate more efficiently, and innovate and experiment more strategically." The Boston Consulting Group , *The Great Software Transformation*, 11 (2013). For example, software is now responsible for 80% of innovation in the automobile industry, just one of many sectors where products are increasingly relying on sophisticated software solutions to improve efficiency, safety, and functionality. Peter Anden et al., *The Perils of Ignoring Software Development*, McKinsey Quarterly (Feb. 2015).

Second, the software industry makes a significant contribution to the nation's investment in research and development ("R&D"). The software industry accounts for 17.2% of the nation's R&D spending, a total of more than $50 billion dollars a year. BSA, *The $1 Trillion Economic Impact of Software*, at 4. BSA member Microsoft alone spends more than $10 billion on R&D each year. Barry Jaruzelski et al., *Global Innovation 1000: Innovation's New World Order*, Strategy and Business (Winter 2015). At an annual growth rate of 13.2%, software is the fastest growing area of R&D

spending in the entire economy. *Id.* Today, U.S. software R&D spending outstrips industrial R&D spending.

Third, the software industry provides nearly 10 million jobs for the American workforce. BSA, *The $1 Trillion Economic Impact of Software*, at 3. That figure includes 2.5 million people directly employed by software companies, which pay salaries far above the national average. *Id.* Indeed, software developers earned an average $108,760 in 2014—more than twice the national average of non-software workers. *Id.* BSA has projected that these high-paying software jobs will grow at a rate of 3.1% through 2020. BSA, *Powering the Digital Economy: A Trade Agenda to Drive Growth*, 4 (2014).

Additionally, software acts as a strong "employment multiplier," given that every two jobs in software support an additional job in other industries. Robert J. Shapiro, *The U.S. Software Industry: An Engine for Economic Growth and Employment*, SIIA White Paper, 6-7 (2014). Today, software indirectly supports an estimated 7.3 million jobs outside the software industry. BSA, *The $1 Trillion Economic Impact of Software*, at 3.

B. The software industry requires copyright protection.

Copyright protection is essential to the continued vitality of the software industry. Software is uniquely easy and cheap to copy and redistribute. Indeed, anyone who obtains a copy can instantly create large numbers of additional copies at no cost and easily redistribute those copies without expense. For example, in *United States v. LaMacchia*, 871 F. Supp. 535, 536-37 (D. Mass. 1994), a college student caused over a million dollars in damage by distributing unlicensed software from a college-owned machine available to him.

Because of the ease with which software may be copied and distributed, software authors need copyright protections for their works before they are willing to release those works into commerce. Without such protections, any software product would be pilfered by users and competitors alike as soon as the first copy is released—and the software creator would be unable to recoup the cost of development.

Other types of intellectual property protection, such as patents and trade secrets, are complements to, not substitutes for, copyright protection. For example, trade secret protection is not available for many types of software, such as some web-based technologies that must be distributed as plain text source code. Patents, which are limited to protecting software function, also cover rights that are distinct from copyright, which protects expression. In addition, the high cost and multi-year application process for software patents may be prohibitive for the needs of many software developers, who require the ability to develop and place their products into commerce at the pace of innovation.

For these reasons, copyright law has formed the foundation of the software industry. Since its earliest days, the software industry has used copyright licenses as the primary means for protecting and distributing its products. Emery Simon, BSA, Testimony before the United States House of Representatives Committee on the Judiciary Subcommittee on Courts, Intellectual Property, and the Internet (June 2, 2014). Indeed, in 1969, when IBM first began to offer software separate from hardware, it used copyright licenses to do so. *Id.*

The copyright licensing model has generated many advantages by providing customers with lower prices, more choices, and post-transaction benefits. For example, many software licenses entitle the licensee to patches and other updates for improved functionality and security. Emery Simon, BSA, Testimony before the United States House of Representatives Committee on the Judiciary Subcommittee on Courts, Intellectual Property, and the Internet. This benefits not only the individual customer, but also the entire software ecosystem, by improving security and by reducing complexity and maintenance costs.

II. Software Is Entitled To Copyright Protection.

Software's entitlement to copyright protection is well established. It rests on the 1980 amendment to the Copyright Act, in which Congress formally extended copyright protection to software without special conditions, and decades of subsequent case law recognizing that original software expression is copyrightable as long as generally-applicable copyright principles are satisfied.

A. Congress provided software with copyright protection.

Congress in 1974 established the National Commission on New Technological Uses of Copyrighted Works ("CONTU") to study and compile data on copyright protection for computer programs, among other issues. *See* Pub. L. No. 93-573, § 201, 88 Stat. 1873 (1974); Final Report of the National Commission on New Technological Uses of Copyrighted Works 9 (July 31, 1978) (hereinafter "CONTU Report"). The recommendations in the Commission's final report, issued on July 31, 1978, formed much of the basis for the 1980 amendments to the Copyright Act, which formally included computer software as copyrightable material.

In its discussion regarding the copyrightability of computer software, the CONTU Report repeatedly emphasized that software, like any other work of authorship, should be protected under the Act as long as it is original. The CONTU Report advocated amending the 1976 Copyright Act "to make it explicit that computer programs, to the extent that they embody an author's original creation, are proper subject matter of copyright." CONTU Report at 1. In making that recommendation, the committee rejected the approach of Commissioner Hersey, who would not have given copyright protection to "a computer program in the form in which it is capable of being used to control computer operations." *Id.* Moreover, the CONTU Report eschewed the approach advocated by Commissioner Nimmer, who believed computer programs could be copyrighted "only when their use leads to copyrighted output." *Id.* at 21.

CONTU embraced the rule of Section 102: a copyrightable work is an "original work[] of authorship fixed in any tangible medium of expression." 17 U.S.C. § 102(a). The Commission thus rejected calls to deny a computer program copyright protection simply because it possessed "utilitarian" aspects or because "the words of a program are used ultimately in the implementation of a process." CONTU Report at 21. Instead, "[a]ll that is needed to satisfy both the Constitution and the statute is that the 'author' contributed something more than a 'merely trivial' variation, something recognizably 'his own.'" *Id.* at 25 (citation omitted).

The policy choice recommended by CONTU—that the barriers to copyright protection be minimal for software—was codified by Congress in the 1980 amendments to the Copyright Act. Congress expressly enacted copyright protection for computer programs, defined as "a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result." *See* 17 U.S.C. § 101; Pub. L. No. 96-517, § 10, 94 Stat. 3015, 3028 (1980).

Consequently, computer programs are like any other copyrightable subject matter. If they constitute original expression, *see Feist Publ'ns, Inc. v. Rural Tel. Serv. Co.*, 499 U.S. 340, 345 (1991) ("[t]he *sine qua non* of copyright is originality"), they are entitled to full copyright protection, subject only to the limits imposed by general copyright principles. There are no special restrictions on software copyrightability.

B. Courts have consistently recognized copyright protection for software.

Following the course charted by Congress, courts repeatedly have confirmed that computer software is copyrightable. As this Court has recognized, the Copyright Act "was amended [in 1980] to include expressly what had been generally assumed as implicit in the Act of 1976 itself," which is "that computer programs were proper subjects of copyright." *M. Kramer Mfg. Co. v. Andrews*, 783 F.2d 421, 432 (4th Cir. 1986).

Numerous courts have refused to narrow software's eligibility for copyright protection. For example, in *Apple v. Franklin*, the Third Circuit rejected the contention that operating systems, as opposed to application programs, are not copyrightable because they are "purely utilitarian works." *Apple Comput., Inc. v. Franklin Comput. Corp.*, 714 F.2d 1240, 1250 (3d Cir. 1983) (internal quotation marks omitted). The court found "no reason to afford any less copyright protection to the instructions in an operating system program than to the instructions in an application program." *Id.* at 1251. Drawing an analogy, the court explained that "instructions in an operating system program" merit no less copyright protection than "instructions … written in ordinary English in a manual [describing] the necessary steps to activate an intricate complicated machine." *Id.*

The Ninth Circuit reached a similar conclusion when confronted with the argument that "a computer program is protected under the Copyright Act only if the program embodies expression *which is communicated to the user when the program is run on a computer*." *Apple Comput., Inc. v. Formula Int'l Inc.*, 725 F.2d 521, 524 (9th Cir. 1984). The court held that this position "is contrary to the language of the Copyright Act, the legislative history of the Act, and the existing case law concerning the copyrightability of computer programs." *Id.* It explained that software "embodies expression," and the Copyright Act has never "required that the expression be communicated to a particular audience." *Id.* at 525.

This Court has also read the Copyright Act broadly, holding that certain elements of video games are protectable as audiovisual works. *M. Kramer*, 783 F.2d at 435. As the Court explained, even if "the idea of the game, itself is not protected," "the shape and characteristics of the cards and the shapes, sizes, colors, sequences, arrangements and sounds[, which] provide[] something new or additional over the idea" are protected. *Id.* (internal quotation marks omitted). It concluded that it is "unquestionable that video games in general are entitled to copyright protections as audiovisual works." *Id.* at 436 (internal quotation marks omitted).

Courts have applied to software the general rule that "copyright protection extends beyond a literary work's strictly textual form to its non-literal components." *Comput. Assocs. Int'l, Inc. v. Altai, Inc.*, 982 F.2d 693, 701 (2d Cir. 1992). Accordingly, they have routinely held that copyright protection "extends not only to the 'literal' elements of computer software—the source code and object code— but also to a program's nonliteral elements, including its structure, sequence, organization, user interface, screen displays, and menu structures." *Gen. Universal Sys., Inc. v. Lee*, 379 F.3d 131, 142 (5th Cir. 2004) (internal footnote omitted); *see also Oracle*, 750 F.3d at 1355-56.

This understanding stems from the general recognition that "the programmer's choice of program structure and design may be highly creative and idiosyncratic." *Sega Enters. Ltd. v. Accolade, Inc.*, 977 F.2d 1510, 1524 (9th Cir. 1992). Thus, choices regarding, for example, a program's "organization of inter-modular relationships, parameter lists, and macros" may be copyrightable non-literal elements to the extent they embody "expression of original ideas, as distinguished from the ideas themselves." *Altai*, 982 F.2d at 702-03 (internal quotation marks omitted).

C. To determine the extent to which a particular software program is copyrightable, a court must differentiate between its protected and unprotected elements.

Copyright law protects only "the expression of the idea—not the idea itself." *Mazer v. Stein*, 347 U.S. 201, 217-18 (1954). As Justice Holmes explained, copyright does not extend to "the ideas, as distinguished from the words in which those ideas are clothed." *Kalem Co. v. Harper Bros.*, 222 U.S. 55, 63 (1911). This limiting principle is today codified at 17 U.S.C. § 102(b), which states that "[i]n no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work." 17 U.S.C. § 102(b).

Whether copyright's protections have been infringed is easy to determine in cases of verbatim copying. When a software program or elements thereof are copied, copyright protection is obviously implicated (unless certain exceptions apply, such as merger or scenes a faire).

Cases of non-verbatim copying, however, must be analyzed in terms of an "idea/expression dichotomy." *See*, *e.g.*, *Altai*, 982 F.2d at 703; *Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc.*, 797 F.2d 1222 (3d Cir. 1986); *Franklin*, 714 F.2d at 1252. And courts have long recognized that "[d]rawing the line between idea and expression is a tricky business." *Altai*, 982 F.2d at 704; *see also Whelan*, 797 F.2d at 1235 ("the line between idea and expression is elusive"). As Judge Learned Hand stated a century ago, "it has never been very satisfactorily established, and probably never can be, at what point a plagiarism ceases to copy the expression of an author's ideas and steals only the ideas themselves." *Fitch v. Young*, 230 F. 743, 745-46 (S.D.N.Y. 1916).

The task of separating idea from expression is not amenable to bright-line rules. *See Peter Pan Fabrics, Inc. v. Martin Weiner Corp.,* 274

F.2d 487, 489 (2d Cir. 1960) ("[o]bviously, no principle can be stated as to when an imitator has gone beyond copying the 'idea,' and has borrowed its 'expression'"). Consequently, the analysis must "inevitably be *ad hoc*." *Id.*

In place of bright-line rules, courts have utilized analytical frameworks to draw the necessary distinction called for by § 102(b). The Third Circuit, for example, suggested that a computer program's "idea" is its underlying "purpose or function" and that its expression is "everything that is not necessary to that purpose or function." *Whelan*, 797 F.2d at 1236 (emphasis omitted). Thus, "'approximation' of a program short of perfect reproduction" is still infringement. *Id.* at 1237.

Applying this principle, *Whelan* held that the "purpose" of the software at issue "was to aid in the business operations of a dental laboratory" and that "the structure of the program was not essential to that task," as there were "other programs on the market … that perform[ed] the same functions but ha[d] different structures and designs." *Whelan*, 797 F.2d at 1238. Accordingly, *Whelan* found copyrightable expression and infringement. *Id.*

The Second Circuit later proposed an alternative to *Whelan*'s approach with respect to the copyrightability of software's non-literal elements. *Altai*, 982 F.2d at 702-03. Although the Circuit agreed that "those elements of a computer program that are necessarily incidental to its function are … unprotectable," it disagreed that each computer program contains "only one 'idea', in copyright law terms." *Id.* at 705 (internal quotation marks omitted). Accordingly, it suggested a three-part test by which a court would examine the various expressions and ideas in a program at different levels of abstraction, and thereby determine which aspects are protected under § 102(a) and which are excluded under § 102(b):

1. **Abstraction**. In the abstraction step, "a court should dissect the allegedly copied program's structure and isolate each level of abstraction contained within it," effectively "retrac[ing] and map[ping] each of the designer's steps." *Id.* at 707. This step considers the designer's choices, "begin[ning] with the code and end[ing] with an articulation of the program's ultimate function." *Id.*

2. **Filtration.** In the filtration step, the court proceeds by "examining the structural components at each level of abstraction to determine whether their particular inclusion at that level was 'idea' or was [1] dictated by considerations of efficiency, so as to be necessarily incidental to that idea;[2] [2] required by factors external to the program

[2] The court reasoned that such aspects are precluded by the merger doctrine—i.e., "[w]hen there is essentially only one way to express an idea,

itself;[3] or [3] taken from the public domain and hence is nonprotectable expression." *Id.*

3. **Comparison.** Finally, the comparison step calls for the court to determine whether the defendant copied any of the remaining protectable expression. *Id.* at 710-11.

Applying this "abstraction-filtration-comparison" framework, *Altai* identified five levels of abstraction for the allegedly infringing program: object code, source code, parameter lists, services, and general organization. *Id.* at 714. The court found no similarity in the object or source codes and insufficient similarity among the parameter lists, other than that dictated by technical demands or already in the public domain. *Id.* at 714-15. It also found that the services were "determined by the demands of the operating system," and that the organization "follow[s] naturally from the work's theme rather than from the author's creativity" (*i.e.*, scenes a faire). *Id.* (internal quotation marks omitted). Accordingly, the court found no infringement in that case.

---

the idea and its expression are inseparable and copyright is no bar to copying that expression." *Altai*, 982 F.2d at 707-08.

[3] The court reasoned that such aspects are precluded by the scenes a faire doctrine—*i.e.*, where "it is virtually impossible to write a program to perform particular functions in a specific computing environment without employing standard techniques." *Altai*, 982 F.2d at 709.

The Fifth, Ninth, Tenth, and Eleventh Circuits have all adopted the Second Circuit's tri-partite framework. *See Comput. Mgmt. Assistance Co. v. Robert F. DeCastro, Inc.*, 220 F.3d 396, 400 (5th Cir. 2000) ("We use the "abstraction-filtration" method to determine copyright protection."); *Sega Enters. Ltd.*, 977 F.2d at 1525 ("the Second Circuit's approach is an appropriate one"); *Gates Rubber Co. v. Bando Chem. Indus., Ltd.*, 9 F.3d 823, 834 (10th Cir. 1993) ("we adopt the 'Abstraction–Filtration–Comparison test"); *Bateman v. Mnemonics, Inc.*, 79 F.3d 1532, 1545 (11th Cir. 1996) ("such an analysis is necessary").

Applying Ninth Circuit law, the Federal Circuit recently employed the three-step framework to confirm the copyrightability of the application programming interface ("API") of software owned by Oracle. *Oracle*, 750 F.3d at 1357. In that case, Oracle distributed a library of pre-written computer programs, organized hierarchically into "packages," for use by Java developers. *Id.* at 1347-49. The packages provided commonly-used functions, each of which a developer's program could invoke by referencing the appropriate package and the function therein by its "interface"—*i.e.*, the function's name and necessary parameters. *Id.* The functions' interfaces were declared within each package using "declaring code." *Id.*

The Court invoked the "abstraction-filtration-comparison" framework and found that, based on the facts of that case, "the declaring code and the structure, sequence, and organization of the 37 Java API packages are entitled to copyright protection." *Id.* at 1354. First, the court determined that the defendant (Google) copied both the declaring code, and the collective "structure, sequence, and organization" of the packages. *Id.* at 1359-68. Second, applying the filtration step, the court found that both elements were protectable expression because neither was dictated by efficiency or technical considerations, such that the merger or scenes a faire doctrines would bar copyrightability. *Id.* The court further found that the short phrases doctrine did not bar copyrightability of the interfaces, which were collections of such phrases. *Id.* at 1362. The court thus concluded that Oracle's software included protectable expression at two levels, and that Google copied that expression.

These examples show that, although a computer program is a functional work, that fact by itself does not deprive it of the protection of the Copyright Act, and "the court must nevertheless determine whether it contains any separable expression entitled to protection." *Id.* at 1367. "Copyrighted software ordinarily contains both copyrighted and unprotected or functional elements," *Sony Computer Entm't, Inc. v. Connectix Corp.*, 203

F.3d 596, 599 (9th Cir. 2000), and the fact that both elements coexist in the same work does not strip the copyrightable elements of protection. Instead, the court's task is to "ferret out apparent expressive aspects of a work," as distinct from "unprotectable ideas, facts, processes, and methods of operation." *Oracle*, 750 F.3d at 1357. The abstraction-filtration-comparison framework is one well-established approach for accomplishing this goal.

III. The District Court Did Not Properly Analyze The Copyrightability Of The SAS System.

In this case, the copyrighted work (the SAS System) is a computer program that reads input expressed in a particular format (SAS Language), performs the functions specified by that input, and outputs the results in a particular format. *SAS Inst. Inc. v. World Programming Ltd.*, 64 F. Supp. 3d 755, 762 (E.D.N.C. 2014). The input must be formatted in accordance with the SAS language, and it may invoke any number of pre-written functions, known as "SAS Procedures" or "PROCs," through pre-defined interfaces. *Id.* at 762, 778.

There is no dispute that WPL copied all of these aspects. WPL's software recognizes the same inputs as SAS's system, provides most (if not all) the same PROCs through the same interfaces, recognizes the same parameters, and produces the same or substantially similar output. *Id.* at 764. The question is only whether any of these elements are copyrightable.

SAS accuses WPL of infringing SAS's copyrights in, among other things, the PROCs and associated parameters made available by the SAS system. *Id.* at 775. The district court held that those elements are not copyrightable, stating that SAS sought "to copyright the idea of a program which interprets and compiles the SAS Language," when "copyright law provides no protection to ideas." *Id.* at 776.

But the district court's analysis was conclusory, falling far short of the fact-intensive inquiry the copyrightability standard requires. The court failed to distinguish between the copyrightability and infringement inquiries, it did not acknowledge that SAS software includes both literal or non-literal elements that may be entitled to protection, and it did not apply any framework for distinguishing the ideas embodied in those elements from their expression.

Indeed, the court did not recognize the possibility that some aspects of the SAS system *could* be copyrightable expression, even if other aspects were not. Instead, the court simply regarded the entire SAS system—including the "SAS Procedures"—as incidental to the SAS Language, which it regarded as *per se* uncopyrightable. *SAS Institute*, 64 F. Supp. 3d at 762, 775-76.

The district court's analysis with respect to the copyrightability of the SAS PROCs was insufficient. *Id.* As the court recognized, a SAS program may invoke certain functionality by calling named SAS PROCs. *Id.* at 762. But the court did not consider whether the selection of available procedures, their interfaces, their organization, or any other qualities might constitute protectable expression. Instead, the court simply characterized these PROCs as "elements of the SAS language," and held them uncopyrightable on the ground that language elements are, *per se*, outside the copyright law. *Id.* at 775-76.

The SAS PROCs "are made up of various functions, routines, statements, formats, engines, macros, procedures, and options." *Id.* at 775. But the district court did not analyze whether any of these elements constituted protectable expression. For example, the court performed no abstraction analysis to identify potentially copyrightable expressions at different levels. It performed no filtration analysis to determine whether these expressions were "necessarily incidental to the [underlying] idea," "required by factors external to the program," or "taken from the public domain." *Altai*, 982 F.2d at 707. The district court thus erred by making no attempt to "ferret out apparent expressive aspects" of SAS's software. *Oracle*, 750 F.3d at 1357.

The district court's opinion stands in sharp contrast to the Federal Circuit's *Oracle* decision, which illustrates how the naming, selection, compilation, or organization of functional interfaces, such as SAS's PROCs, may be copyrightable when properly analyzed. 750 F.3d at 1348. The district court here did not arrive at its conclusion after identifying the various abstractions of the protected software or testing whether various copyright doctrines, such as merger or scenes a faire, apply. Instead, its brief, three-page analysis simply equated the entire SAS software system with the "idea" of a programming language. *SAS Institute*, 64 F. Supp. 3d at 776.

The district court distinguished *Oracle* on the basis that "[t]he declaring code at issue in *Oracle* was not an element of the … language itself," while the PROCS (evidently) were part of the SAS language itself. *Id.* at 777. But the district court never explained what it means for functionality to be "an element of the … language itself" or how SAS PROCs meet that definition as distinguished from *Oracle*'s Java packages.

In sum, the district court did not even begin to answer the central question of copyrightability, which is whether the PROCs include protectable expression, separate from their function. That protectable expression might be found in literal elements (such as the PROCs' interfaces) or in non-literal elements (such as the organizational choice of collecting those procedures into a single system). Because the district court considered neither possibility, its decision should be vacated and the case remanded to allow the district court to conduct the proper analysis. That analysis may include clarifying exactly what aspects of the PROCs SAS believes constitute protectable expression, applying a rigorous analytical framework (such as the abstraction-filtration-comparison test) to properly determine whether SAS's functions comprise protectable expression, and determining whether any affirmative defenses—such as merger, fair use, or scenes a faire—might apply.

CONCLUSION

The Court should vacate and remand to allow the district court to apply in the first instance the proper analytical framework for assessing copyrightability and infringement.

Respectfully submitted,

/s/ Andrew J. Pincus
Andrew J. Pincus
Paul W. Hughes
Jonathan Weinberg
MAYER BROWN LLP
1999 K Street, N.W.
Washington, DC 20006-1101
(202) 263-3000

December 27, 2016

CERTIFICATE OF COMPLIANCE

Pursuant to Federal Rule of Appellate Procedure 32(a)(7)(C), the undersigned counsel for Defendant-Appellant certifies that this brief:

(i) complies with the type-volume limitation of Rule 32(a)(7)(B) because it contains 5,001 words, including footnotes and excluding the parts of the brief exempted by Rule 32(a)(7)(B)(iii); and

(ii) complies with the typeface requirements of Rule 32(a)(5) and the type style requirements of Rule 32(a)(6) because it has been prepared using Microsoft Office Word 2007 and is set in Century Schoolbook font in a size equivalent to 14 points or larger.

Dated: December 27, 2016

/s/ Andrew J. Pincus
Andrew J. Pincus

CERTIFICATE OF SERVICE

I hereby certify that that on December 27, 2016, I electronically filed the foregoing Brief of BSA | The Software Alliance As *Amicus Curiae* In Support Of Plaintiff-Appellant with the Clerk of the Court using the appellate CM/ECF system. I further certify that all participants in this case are registered CM/ECF users and that service will be accomplished via CM/ECF.

Dated: December 27, 2016

/s/ Andrew J. Pincus
Andrew J. Pincus