No. 2021-1542

IN THE
United States Court of Appeals for the Federal Circuit

SAS INSTITUTE, INC.,

Plaintiff-Appellant,

v.

WORLD PROGRAMMING LIMITED,

Defendant-Appellee.

On Appeal from the United States District Court for the Eastern District of Texas
Case No. 2:18-cv-00295-JRG, Hon. J. Rodney Gilstrap, Chief Judge

BRIEF OF 54 COMPUTER SCIENTISTS IN SUPPORT OF APPELLEE AND AFFIRMANCE

Jef Pearlman
INTELLECTUAL PROPERTY & TECHNOLOGY LAW CLINIC
UNIVERSITY OF SOUTHERN CALIFORNIA GOULD SCHOOL OF LAW
699 Exposition Blvd.
Los Angeles, CA 90089-0071
(213) 740-7088
jef@law.usc.edu

Counsel for Amici Curiae

UNITED STATES COURT OF APPEALS FOR THE FEDERAL CIRCUIT

CERTIFICATE OF INTEREST

Case Number	2021-1542
Short Case Caption	SAS Institute, Inc. v. World Programming Limited
Filing Party/Entity	54 Computer Scientists (see Attachment A for list)

Instructions: Complete each section of the form. In answering items 2 and 3, be specific as to which represented entities the answers apply; lack of specificity may result in non-compliance. **Please enter only one item per box; attach additional pages as needed and check the relevant box**. Counsel must immediately file an amended Certificate of Interest if information changes. Fed. Cir. R. 47.4(b).

I certify the following information and any attached sheets are accurate and complete to the best of my knowledge.

Date: 08/30/2021

Signature: /s/Jeffrey Theodore Pearlman

Name: Jeffrey Theodore Pearlman

1. Represented Entities. Fed. Cir. R. 47.4(a)(1).	**2. Real Party in Interest.** Fed. Cir. R. 47.4(a)(2).	**3. Parent Corporations and Stockholders.** Fed. Cir. R. 47.4(a)(3).
Provide the full names of all entities represented by undersigned counsel in this case.	Provide the full names of all real parties in interest for the entities. Do not list the real parties if they are the same as the entities. [✓] None/Not Applicable	Provide the full names of all parent corporations for the entities and all publicly held companies that own 10% or more stock in the entities. [✓] None/Not Applicable
54 Computer Scientists (see Attachment A for list)		

[✓] Additional pages attached

4. Legal Representatives. List all law firms, partners, and associates that (a) appeared for the entities in the originating court or agency or (b) are expected to appear in this court for the entities. Do not include those who have already entered an appearance in this court. Fed. Cir. R. 47.4(a)(4).

☑ None/Not Applicable ☐ Additional pages attached

5. Related Cases. Provide the case titles and numbers of any case known to be pending in this court or any other court or agency that will directly affect or be directly affected by this court's decision in the pending appeal. Do not include the originating case number(s) for this case. Fed. Cir. R. 47.4(a)(5). See also Fed. Cir. R. 47.5(b).

☑ None/Not Applicable ☐ Additional pages attached

6. Organizational Victims and Bankruptcy Cases. Provide any information required under Fed. R. App. P. 26.1(b) (organizational victims in criminal cases) and 26.1(c) (bankruptcy case debtors and trustees). Fed. Cir. R. 47.4(a)(6).

☑ None/Not Applicable ☐ Additional pages attached

ATTACHMENT A

List of Computer Scientist Amici Curiae (in alphabetical order):

1. Dr. Harold Abelson
2. Jon Bentley
3. Matthew Bishop
4. Joshua Bloch
5. Gilad Bracha
6. Daniel Bricklin
7. Frederick P. Brooks, Jr.
8. R.G.G. Cattell
9. David Clark
10. William Cook
11. Thomas H. Cormen
12. Miguel de Icaza
13. Dr. L Peter Deutsch
14. Whitfield Diffie
15. David L. Dill
16. Dawson Engler
17. Bob Frankston
18. Neal Gafter
19. Erich Gamma
20. Andrew Glover
21. Allan Gottlieb
22. Robert Harper
23. Maurice Herlihy
24. Tom Jennings
25. Alan Kay
26. Brian Kernighan
27. David Klausner
28. Kin Lane
29. Ed Lazowska
30. Douglas Lea
31. Bob Lee
32. Harry Lewis
33. Douglas McIlroy
34. Paul Menchini
35. James H. Morris
36. Peter Norvig

37. Martin Odersky
38. David Patterson
39. Tim Peierls
40. Curtis Schroeder
41. Robert Sedgewick
42. Mary Shaw
43. Alfred Z. Spector
44. Michael Stonebraker
45. Ivan E. Sutherland
46. Andrew Tanenbaum
47. Brad Templeton
48. Andries van Dam
49. Guido van Rossum
50. John Villasenor
51. Jan Vitek
52. James Waldo
53. Dan Wallach
54. Frank Yellin

TABLE OF CONTENTS

TABLE OF AUTHORITIES

INTEREST OF AMICI CURIAE[1]

Amici are 54 computer scientists, engineers, and computer science professors who are pioneering and influential figures in the computer industry.[2] Amici include the architects of iconic computers including the IBM S/360; languages such as AppleScript, AWK, C, PL/I, Python, Scala, Scheme, Standard ML, and Smalltalk. Amici are responsible for key advances in the field, including in computer graphics, computer animation, computer system architecture, cloud computing, algorithms, public key cryptography, object-oriented programming, relational databases, design patterns, virtual reality, and the spreadsheet. Amici wrote the standard college textbooks in areas including artificial intelligence, algorithms, computer architecture, computer graphics, computer security, functional programming, Java programming, operating systems, software engineering, and the theory of programming languages.

Amici are widely recognized for their achievements. They include 4 Association for Computing Machinery (ACM) Turing Award recipients (computer science's most prestigious award); 17 ACM Fellows; 7 Institute of Electrical and

[1] No party or party's counsel authored this brief in whole or in part or contributed money that was intended to fund preparing or submitting this brief. No one other than amicus and its counsel contributed money that was intended to fund preparing or submitting this brief. Pursuant to Fed. R. App. P. 29(a)(2), all parties have consented to the filing of this brief.
[2] Short biographies of Amici are provided in the Appendix.

Electronics Engineers (IEEE) Fellows; 4 Computer History Museum (CHM) Fellows; 3 National Academy of Sciences (NAS) Members; 12 National Academy of Engineering (NAE) Members; 3 American Association for the Advancement of Science (AAAS) Members; 9 American Academy of Arts and Sciences (AAoAS) Members; 2 National Medal of Technology recipients; and numerous professors at many of the world's leading universities.

As computer scientists, Amici have long relied on reimplementing languages to create compatible software. They join this brief because they believe, based on their extensive experience with and knowledge of computer software and programming, that the decision below should be affirmed and that reversing it would threaten to upend decades of settled expectations across the computer industry and chill continued innovation in the field.

Amici submit this brief to offer the Court their technical expertise and industry knowledge. Specifically, Amici address some of the arguments and technical assertions made by the parties and other amici, including the three computer scientists supporting SAS (*Williams, Layman, and Sherriff* in the *Williams, Layman, and Sherriff Brief*).[3]

[3] A number of Amici previously submitted amicus briefs to the Supreme Court in its recent consideration of *Google v. Oracle*. In that case, at issue was the copyrightability of Application Programming Interfaces (APIs), which extend the Java language, and Google's fair use in copying those APIs. Much of the

SUMMARY OF ARGUMENT

Plaintiff SAS Institute, Inc. (SAS) alleges that World Programming Limited (WPL) has infringed copyrights covering the "SAS System." While much of this appeal concerns the burden of proof for what is protectable in that "system," the Court should not lose sight of the underlying allegations; even as described by SAS, nothing protectable has been copied.

What SAS calls the "SAS System" encompasses a wide variety of materials and ideas. There is no dispute that some are copyrightable, while others are not. Critically, while the SAS System includes a copyrightable software implementation of a programming language, SAS does not allege that WPL copied *any* of this software. Instead, it alleges that WPL unlawfully copied what SAS calls "input formats" and "output designs." These terms, however, mislead more than they inform. At their core, they are merely synonyms for a programming language and the output of user-designed software using that language—the "specifications" for that language. Neither is copyrightable.

---

additional explanation provided in that brief applies in this case. If anything, the "formats" at issue here are a core part of the SAS language and therefore even further from copyrightable subject matter than the APIs in *Google v. Oracle*. For additional detail, Amici refer this Court to the *Brief Amici Curiae of Eighty Three Computer Scientists* (Jan. 13, 2020), *Google LLC v. Oracle Am., Inc.*, 141 S. Ct. 1183 (2021), https://perma.cc/H8JE-C5NT.

Since software was invented, developers have been writing code that is compatible with existing products. This has, from the start, included writing code that understands existing programming languages. One party may design a programming language and write software—the "implementation"—that understands that language. But other parties have historically been free to write their *own* "reimplementations" that understand the same language. Here, SAS admits that the language is free for anyone to *write* code in, but asserts that only SAS can offer software that *reads* and *understands* that code. But a language is not a copyrightable work, and it certainly makes no sense for a language to be uncopyrightable for writing but copyrightable for reading.

SAS also asserts that if anyone else writes software that understands that language, they infringe by producing the same output that the SAS System produces. This is similarly untenable. The outputs of a program are determined by the code written by the *user*, not by SAS, and by the uncopyrightable system of rules for the language it uses. To the extent there is creativity in the output in those programs, it belongs almost exclusively to the party who wrote them—and that party is not SAS.

SAS's positions are incompatible with the law and norms that have driven the software revolution. This Court should affirm the decision below and preserve one of the major features that have made the software industry so vibrant for the past half-century.

ARGUMENT

The court below correctly concluded that SAS had failed to meet its burden to show that any protectable element of the SAS System was copied. This is more than a failure to meet a burden, though. As a matter of law, no copyrightable element was actually copied. The "input formats" and "output designs" allegedly copied by WPL are uncopyrightable ideas, while the software implementing those ideas was not copied.

What SAS calls "input formats" and "output designs" are known in the industry as the input and output "specifications" or "interfaces" of a language. The software development industry has long understood and relied on the proposition that these specifications are free to reimplement for those wishing to offer a competing, compatible product. This has spurred innovation across the industry. The Supreme Court recently decided a related issue in a manner that permitted reimplementation, consistent with 50 years of this industry practice. *See Google LLC v. Oracle Am., Inc.*, 141 S. Ct. 1183 (2021). This Court should do the same by affirming the decision below. A contrary result could upend this longstanding driver of competition and innovation.

I. THE INDUSTRY HAS LONG RECOGNIZED THAT REIMPLEMENTION OF SPECIFICATIONS IS PRO-INNOVATION AND NONINFRINGING.

SAS's fundamental complaint is that WPL "create[d] a clone of the SAS Software," to compete with SAS. *SAS Brief* at 1 (quotation omitted). But independent, lawful development of competitive, compatible, drop-in replacements for software *without* copying that software has been a part of the marketplace since there *was* a software marketplace. Programming languages, in particular, are often the subject of this type of competition, which is critical to maintaining that robust marketplace.

A. A language's specification is an idea while its implementation is software.

To understand this case, it is critical to recognize and keep in mind the difference between a language's *specification* and *implementation*. The specification tells developers how to write code that will successfully execute, as well as what that code will do when it executes. It tells both users and implementers what commands the system offers, what additional information needs to be provided with those commands, and what the results should be. It is a set of ideas. Those ideas may be communicated through a written manual or a web page, but they are neither a "work" in a copyright sense nor a "set of statements" to be executed by a computer. *See* 17 U.S.C. § 101. They are a shared set of rules that allow programmers to use the language or to write new implementations.

The implementation is the software that takes programs written in the appropriate language and executes them. The implementer, like the users of the language, must know and conform to the specification. But the implementation is *actual* software that is executed by a computer. The output of the software is, for the most part, dictated by the specification and the input provided by the user. There are many ways for the implementation to produce the correct result. But if the implementer has done its job, then the same input code will always produce essentially the same results. The implementation is, generally, copyrightable.

There appears to be no dispute here that WPL has written its own implementation, or that it did so "from scratch, using a different programing language and employing a unique 'sequence, structure, and organization.'" *WPL Brief* at 9 (record citations omitted). That software understands the same input formats—that is, the same language—as the software written by SAS. This is called "reimplementation" and is part and parcel of lawfully creating competitive software. The overlap between the original implementation and the reimplementation is solely the *functionality* of software, including what types of inputs it understands and what output it produces. These are not the subject of copyright.

B. Industry practice has always supported reimplementing a specification.

Copyright's protection of implementations but not ideas allows developers to create competitive, compatible software. This capability has driven the software industry since its inception. There are myriad examples; this table presents just a few, and * indicates examples that are also programming languages:

Interface	**Creator**	**Year**	**Reimplementer**	**Year**
FORTRAN*	IBM	1958	Univac	1961
IBM S/360 ISA	IBM	1964	Amdahl Corp.	1970
C*	AT&T / Bell Labs	1976	Mark Williams Co.	1980
Unix	AT&T / Bell Labs	1976	Mark Williams Co.	1980
VT100 Escape Sequences	Digital Equipment	1978	Heathkit	1980
IBM PC BIOS	IBM	1981	Phoenix Technologies	1984
MS-DOS CLI	Microsoft	1981	FreeDOS Project	1998
Hayes modem command set	Hayes Micro	1982	Anchor Automation	1985
PostScript*	Adobe	1985	Aladdin Enterprises	1988
SMB	Microsoft	1992	Samba Project	1993
Win32	Microsoft	1993	Wine Project	1996
Java*	Sun	1998	Google/Android	2008
Delicious web API	Delicious	2003	Pinboard	2009

Bloch, Joshua J., *A Brief, Opinionated History of the API*, Proceedings of the Companion Publication of the 2014 ACM SIGPLAN Conference on Systems, Programming, and Applications: Software for Humanity (2014). *See also* Dennis Ritchie*,* Reply to alt.folklore.computers Usenet Post *Coherent* (Apr. 10, 1998), https://perma.cc/Y7XC-9YWE (describing AT&T's visit to the developer of a UNIX reimplementation and decision to move on after finding no evidence *code* was copied).

Williams, Layman, and Sherriff argue that "WPL is free to create a competing program that offers similar statistical tools. Other companies have, including IBM and Microsoft." *Williams, Layman, and Sherriff Brief* at 5. This is true, but irrelevant. That other companies have chosen not to reimplement a platform doesn't indicate that doing so is impermissible. There are myriad word processors possessing many different document formats, but many of them can read or write formats "belonging" to the others. Each individual developer decides for themself whether or not to reimplement preexisting interfaces based on a variety of factors unrelated to copyright law.

What *Williams, Layman, and Sherriff* refer to as "knockoff product[s]," *id. at 5*, are critical to the software development ecosystem. They permit developers to leverage the knowledge and experience they have writing a particular language when they move to a new system offered by another party—a system that may be faster, more powerful, or run on different devices. Forbidding such compatibility would result in "lock-in," where developers are unable to move to other, potentially better platforms because copyright law demands they learn new ways to do the same things.

To the contrary, courts have recognized benefits of reimplementation for decades, most recently in *Google v. Oracle*:

> The record here demonstrates the numerous ways in which reimplementing an interface can further the development of computer programs. The jury heard that shared interfaces are necessary for different programs to speak to each other. It heard that the reimplementation of interfaces is necessary if programmers are to be able to use their acquired skills. It heard that the reuse of APIs is common in the industry.

Google LLC v. Oracle Am., Inc., 141 S. Ct. at 1203–04. *See also Lotus Dev. Corp. v. Borland Int'l, Inc.*, 49 F.3d 807, 821 (1st Cir. 1995), aff'd, 516 U.S. 233 (1996) (Boudin, J., concurring) ("it is hard to see why customers who have learned the Lotus menu and devised macros for it should remain captives").

Williams, Layman, and Sherriff argue that "[w]hen a competitor can merely duplicate the designs of an existing product, there is no incentive to make a better version of the software." *Williams, Layman, and Sherriff Brief* at 26. This is untrue. There are many ways, as the Supreme Court put it, that reimplementation "can further the development of computer programs." *Google v. Oracle*, 141 S. Ct. at 1203. Resources can be devoted to developing better products rather than making up new names for old things. These competing, compatible products may then be faster, have additional features, produce more accurate results, run on different hardware or operating systems, or have numerous other advantages. Thus, such software does not, as they say, "tr[y] to do the same thing in the same way." *Id.* at 30. It tries to do a *compatible* thing in a *better* way.

On the other hand, *forbidding* reimplementation can often remove the motivation to improve. Because it locks in developers and users, there is little incentive to improve the original product. Imagine if the first automobile manufacturer retained a copyright-length monopoly on the layout of the car, the shape of the steering wheel, and the location of the accelerator and brake pedals. It is hard to believe this would have led to a better and more competitive car industry.

The ability to compete for existing users by reimplementing software has created a virtuous cycle in which competitors drive each other to improve. Entire software ecosystems form around successful interfaces, leading to even more innovative products and capabilities. The result is software and computing hardware that has become vastly more powerful, flexible, and capable over the nearly 70 years they have existed.

II. THE "SAS SYSTEM" IS A COMBINATION OF UNCOPYRIGHTABLE IDEAS AND COPYRIGHTABLE SOFTWARE THAT WASN'T COPIED.

SAS obfuscates the scope of its infringement allegations by repeatedly referring to its "SAS System" as a "computer program," *e.g.*, *SAS Brief* at 1, 9, 52, and arguing that as a whole it is creative, original, and copyrighted. *E.g.*, *id.* at 2. *Williams, Layman, and Sherriff* similarly describe the SAS System as a "proprietary computer program." *Williams, Layman, and Sherriff Brief* at 3. Both of these are misleading, hiding the complexity of what the SAS System actually is.

What SAS calls the SAS System is a *platform* consisting of a broad set of features. These include not just actual computer programs written by SAS (like compilers and libraries), but "non-literal elements, namely the SAS System's input formats, output designs, and naming and syntax." Appx6. Despite describing the System as a computer program, even *Williams, Layman, and Sherriff* actually recognize that the input formats are *not* part of a program. *See Williams, Layman, and Sherriff Brief* at 4 ("Input formats are the way a user provides instructions *to the software*, to guide the mathematical calculations and statistical analyses that the user needs . . . .") (emphasis added). This is a *specification*, not a program.

And while the SAS System likely contains some copyrightable elements—like the source code that implements the computer programs within the System—it also contains many uncopyrightable elements. Most or all of the purported "works" at issue in this case—and certainly the "input formats" and the core aspects of the "output designs"—are well-understood in the industry to be free to replicate. The "input formats" are merely a programming language, free for all to use. And the outputs, or "output formats," are primarily determined by the user-written programs, not by SAS. As to the implementing code or "computer program" implementing these ideas, it is Amici's understanding that it is undisputed that nothing was copied. *See WPL Brief* at 44 (citing Appx3318-3319 (15:14-16:7)).

III. WHAT SAS AND ITS AMICI CALL "INPUT FORMATS" ARE SIMPLY A FREE-TO-USE PROGRAMMING LANGUAGE.

SAS defines what it refers to as "input formats" as "the complex sets of statements designed by SAS, using keywords selected and arranged by SAS, that are used by the SAS System to carry out statistical analysis, along with their organization." *SAS Brief* at 2. This describes nothing more than the specification for a programming language. And it has long been understood by the industry reflected in industry practice that programming languages are free to reimplement.

A. The "input formats" are a programming language.

SAS repeatedly describes its "input formats" by describing a programming language and its interfaces. For example, in addition to the quote above:

- "Each line involves different 'Statements,' some of which also have 'Options' that 'control … different capabilit[ies] of the procedure.'" *Id.* at 13. This is merely describing statements, operators, functions, and parameters, all of which are basic building blocks of programming languages.
- "Each Procedure is separately written and has its own design including its own syntax, options, statements, and defaults." *Id.* at 14 (quotation omitted). The "design" referred to is just the language itself. The "separately written" procedure is the implementation—which was not copied here. Again, SAS

merely describes how programming languages and the compilers or interpreters that understand them work.

Williams, Layman, and Sherriff offer a similar explanation, walking through a description of "input formats" and "PROCs" that will sound familiar to computer scientists or anyone who is familiar with *Google v. Oracle*. That explanation is just a tutorial introduction as to what comprises an interface. *See Williams, Layman, and Sherriff Brief* at 8-10.

Older languages called these features "subroutines," "procedures," or "functions," and newer languages call them "methods," but the concept is the same. A named entity that takes input in some prescribed format and produces output in some prescribed format (and/or changes the state of the system in some prescribed fashion). The names and inputs are the language, and the code that actually calculates the results is the implementation. SAS and *Williams, Layman, and Sherriff* simply use different words to describe the same thing, further obfuscating the nature of the copyright claims.

The record and history of this dispute confirm this understanding. "As SAS[] witnesses have testified, 'PROC Steps,' 'global statements' and other elements are 'the language of SAS.'" *WPL Brief* at 49 (citations record omitted). Input formats are the language, not the software.

B. Programming languages like the "input formats" are not copyrightable computer programs.

Williams, Layman, and Sherriff argue that "[i]nput formats easily fit within the Copyright Act's definition of a 'computer program['], which is 'a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.'" *Williams, Layman, and Sherriff Brief* at 15. This is a fundamental technical error. Input formats are part of the *specification* for a programming language. But they are not, in and of themselves, a program: there is no "set of statements," 17 U.S.C. § 101, and they are not used by a computer in order to bring about a result.

Importantly, SAS appears to concede that anyone who wants to is free to write code in the language but simultaneously argues that no one else may write software that interprets that same code. Appx16 ("WPL presented evidence that the SAS Language . . . is open and free for public use."); *SAS Inst. Inc. v. World Programming Ltd.*, 64 F. Supp. 3d 755, 762 (E.D.N.C. 2014), *copyright holding vacated as moot*, 874 F.3d 370 (4th Cir. 2017) ("Anyone can write a program in the SAS Language, and it is undisputed that no license is needed to do so."). But in alleging infringement of the language, SAS uses examples focused on the admittedly-noninfringing users: "SAS[]'s only example of an 'input format' is from a *manual* teaching *users* how to write SAS-Language programs and its putative evidence of 'copying' concerns the *SAS-Language elements in user programs* that WPS 'supports.'" *WPL Brief* at 49.

More importantly, the "language" for writing and implementing is the *same thing*, so it cannot be copyrighted in one instance and unprotectable in the other. The district court in North Carolina addressed this conflict directly in a prior SAS case against WPL:

> In essence, by asking the court to find that defendant's software infringes its copyright through its processing of elements [of] the SAS Language, plaintiff seeks to copyright the idea of a program which interprets and compiles the SAS Language—a language anyone may use without a license. However, copyright law provides no protection to ideas.

SAS v. WPL, 64 F. Supp 3d at 776. The North Carolina court was correct: there is no copyright on the language itself, and *that* is what WPL copied when it wrote its own implementation. And SAS cannot copyright the idea of *using* an uncopyrighted language.

C. Creativity is a red herring in this case, as creativity alone does not create copyright protection.

SAS focuses heavily on arguing that it made creative choices in developing its system. *See SAS Brief* at 9, 13, 20-21, 34, 48. The existence of at least minimal creativity does not appear to be in dispute here. But SAS suggests that if something is creative, it is automatically protected by copyright; this is incorrect.

While creativity is a necessary condition for copyrightability, it is not a sufficient one. Inventions are nearly always creative, but are the subject of patents rather than copyright, even when described in a book: "To give to the author of the book an exclusive property in the art described therein, when no examination of its novelty has ever been officially made, would be a surprise and a fraud upon the public. That is the province of letters-patent, not of copyright." *Baker v. Selden*, 101 U.S. 99, 102 (1879). Discovering laws of nature and other natural phenomena is likewise a creative endeavor, but is neither the subject of copyright nor patent protection: "Laws of nature, natural phenomena, and abstract ideas are not patentable." *Alice Corp. Pty. v. CLS Bank Int'l*, 573 U.S. 208, 216 (2014). Just because something is creative does not mean it is protectable under copyright law.

D. It is irrelevant that there are other ways to create incompatible competitive software.

SAS argues that "there are other ways of creating data analysis software." *SAS Brief* at 21 (capitalization changes omitted). *Williams, Layman, and Sherriff* make the same argument. *See Williams, Layman, and Sherriff Brief* at 20 ("There are many other ways to design input formats"), 24 ("Nothing about the underlying analysis or data requires this particular format."). While technically true, this is both misleading and irrelevant.

First, the availability of alternatives for an uncopyrightable set of "input formats" doesn't render the idea copyrightable. Second, this suggestion results in *incompatible* software. When it comes to building software that existing users can benefit from, it is not truly an "alternative" if it results in *in*compatible software. If the "alternative" is using different commands, then it won't understand code that users write in the language they already know. Similarly, if the "alternative" is having software produce substantially different output for the same input, then the competitive software will be useless. One *must* reuse the freely available language to produce, as WPL has, different software that speaks the same language the developers do.

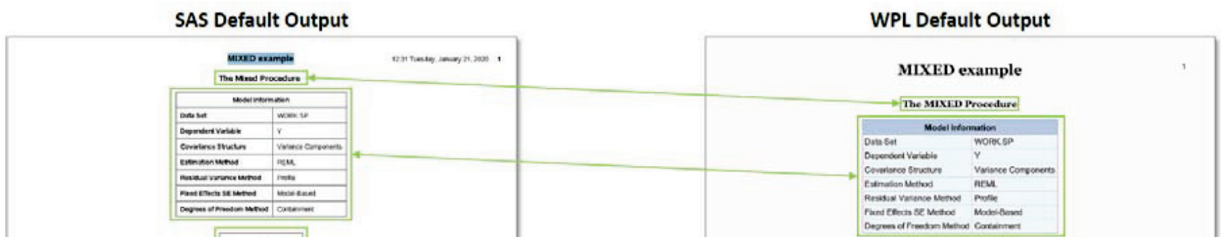
IV. "OUTPUT FORMATS" ARE MERELY THE RESULTS OF EXECUTING THE USER-PROVIDED CODE.

SAS also claims copyright on the "output formats" of its software. *SAS Brief* at 16-24. Like the inputs, the output formats are not software. And to the extent WPL copied these output formats, it appears to have copied only the uncopyrightable elements that are demanded by the functionality of the language. Notably, SAS does not say that the WPS outputs are *the same* as SAS's; instead, it says "that WPS has displays that are '*equivalent*' to SAS's Output Designs with '*similar* graphical output.'" *SAS Brief* at 25 (quoting Appx10) (emphasis added).

This is important because the output formats are not creative expression by the platform developer. They are defined by the results of the PROC—in other words, by what code the user (*not* SAS) has written. The PROC is, in essence, a machine that turns its inputs into its outputs, and if someone else wants to make a compatible "machine" that is usable by programmers trained to operate the first one, it *must* accept the same input formats and generate the same output.

In other words, if a user asks for 2 + 2, the software must produce 4. If the user asks the software to display a table, it must look like a table. If the user asks for a bar graph, then the graph must have bars. There is room for creativity in the implementation of how those things are displayed, but that room is limited. For the rest, it is the user's creativity, and not SAS's, that is relevant.

This is shown by the comparisons made by both SAS and *Williams, Layman, and Sherriff*, portions of which we reproduce here:



MIXED example

12:31 Tuesday, January 21, 2020 1

The Mixed Procedure

Model Information	
Data Set	WORK.SP
Dependent Variable	Y
Covariance Structure	Variance Components
Estimation Method	REML
Residual Variance Method	Profile
Fixed Effects SE Method	Model-Based
Degrees of Freedom Method	Containment

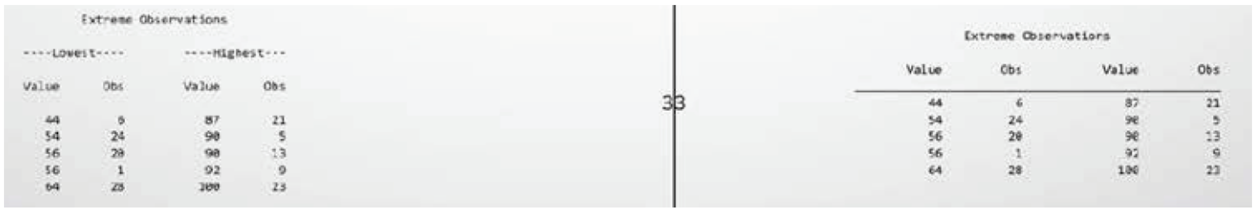
MIXED example

1

The MIXED Procedure

Model Information	
Data Set	WORK.SP
Dependent Variable	Y
Covariance Structure	Variance Components
Estimation Method	REML
Residual Variance Method	Profile
Fixed Effects SE Method	Model-Based
Degrees of Freedom Method	Containment

SAS Brief at 26 (image cropped).



```
        Extreme Observations

----Lowest----      ----Highest---

Value      Obs      Value      Obs

   44        6         87       21
   54       24         90        5
   56       20         90       13
   56        1         92        9
   64       28        100       23
```

33

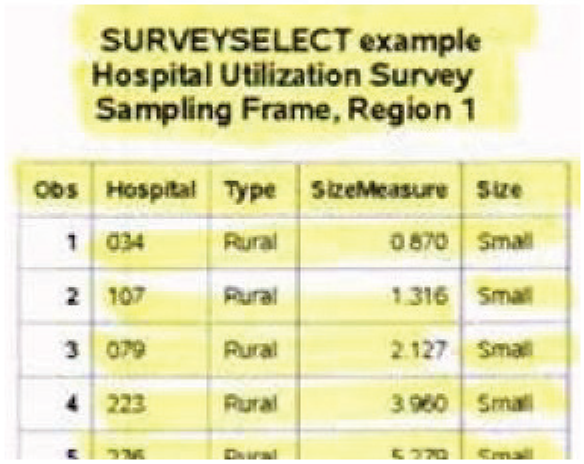
```
                 Extreme Observations

        Value      Obs           Value      Obs
    -------------------------------------------
           44        6              87       21
           54       24              90        5
           56       20              90       13
           56        1              92        9
           64       28             100       23
```

Williams, Layman, and Sherriff Brief at 24 (image cropped).

While the values and names of the output are the same and both are produced in tables—as determined by the user's input—there are significant differences in how the tables are presented. The fonts, lining, spacing, and colors all differ; even the headings are different ("highest" and "lowest" present only on the left). In other words, the only things even arguably copied were those derived from the user's input in accordance with the language's specification.

SAS's own witnesses confirmed that the user specifies "virtually 'all' the[] details" of the output and identified the highlighted portion below as user-defined:



SURVEYSELECT example
Hospital Utilization Survey
Sampling Frame, Region 1

Obs	Hospital	Type	SizeMeasure	Size
1	034	Rural	0.870	Small
2	107	Rural	1.316	Small
3	079	Rural	2.127	Small
4	223	Rural	3.960	Small
5	236	Rural	5.279	Small

WPL Brief at 7-8 (citation omitted and image cropped). Witnesses similarly confirmed that the "[o]utputs are the 'function of the SAS [Language] program that a customer uses and the customer's data'" *Id.* at 53 (citations omitted).

When given the input "2 + 2," a calculator must produce "4" as the output. The same is true for any well-defined computer language. It cannot be the case that copyright allows developers to write software that understands the same input but bars writing software that produces the output the user requested.

CONCLUSION

As explained above, SAS does not dispute that the language is free to use for developers. This language is not software and not a work subject to copyright, and so writing new software that understands the language or produces the same results for the same operations must similarly remain free. To hold otherwise would be to undermine over half a century of practice in the software development industry—practice that has given us the powerful software and vibrant software market we enjoy today. Amici therefore respectfully request that the Court affirm the decision below.

/s/Jeffrey Theodore Pearlman
Jef Pearlman
Counsel for Amici Curiae

August 30, 2021

APPENDIX — LIST OF AMICI

Amici sign this brief on their own behalf, not on behalf of the organizations with which they are affiliated.

1. Dr. Harold Abelson. Professor, MIT. Co-author, innovative introductory CS text with worldwide impact. Founding director, Creative Commons, Public Knowledge. Four major awards for contributions to CS education. Fellow, IEEE.
2. Jon Bentley. Researcher: programming techniques, tools, algorithms. Previously, Distinguished Member of Technical Staff, Bell Labs; Professor, Carnegie-Mellon; visiting faculty, West Point, Princeton.
3. Matthew Bishop. Professor, UC Davis. Author, Computer Security: Art and Science.
4. Joshua Bloch. Professor, Carnegie-Mellon. Specialist in API Design. Previously, Chief Java Architect, Google; Distinguished Engineer, Sun Microsystems. Led design, implementation of numerous Java APIs. Author, Effective Java.
5. Gilad Bracha. Creator Newspeak programming language. Previously, Scientist, Google; VP, SAP Labs; Distinguished Engineer, Sun Microsystems. Co-author, Java Language and VM Specifications. Dahl-Nygaard Prize.

6. Daniel Bricklin. Conceived and co-developed VisiCalc, the first spreadsheet. Fellow, CHM, ACM. Member, NAE. ACM Software System Award, ACM Grace Murray Hopper Award.
7. Frederick P. Brooks, Jr. Professor Emeritus, UNC Chapel Hill. Project Manager, IBM System/360 hardware and OS/360 software. Architect, Stretch and Harvest supercomputers. Founder UNC's CS Department. Author, The Mythical Man-Month. National Medal of Technology, ACM Turing Award. Member, NAS, NAE, British and Dutch academies.
8. R.G.G. Cattell. Distinguished Engineer, Sun Microsystems; Researcher, Xerox PARC, CMU. Responsible for numerous APIs including Enterprise Java, JDBC. Author, first monograph on object/relational databases. Fellow, ACM.
9. David Clark. Internet pioneer. Senior Research Scientist, MIT CSAIL; Technical Director, MIT IPRI. Was Chief Protocol Architect, Internet Activities Board; Chairman, National Academies CSTB. Member, NAE, AAoAS.
10. William Cook. Professor, UT Austin. Chief architect, AppleScript. Dahl-Nygaard Prize.
11. Thomas H. Cormen. Professor, Dartmouth College. Co-author, Introduction to Algorithms. Formerly chair, Dartmouth CS department. ACM Distinguished Educator.

12. Miguel de Icaza. Distinguished Engineer, Microsoft. Cofounder, GNOME, Mono (reimplementing Microsoft's .NET platform on Linux). FSF Software Award, MIT Technology Review Innovator of the Year.
13. Dr. L Peter Deutsch. Co-developed Interlisp-D, Smalltalk-80 at Xerox PARC. Originated Just-In-Time Compilation. Created Ghostscript open-source reimplementation of PostScript. ACM Software System Award. Fellow, ACM.
14. Whitfield Diffie. Discovered public key cryptography, which underlies all modern secure communication. Previously, Chief Security Officer, Sun Microsystems; Manager, Secure Systems Research, Bell-Northern Research. ACM Turing Award. Member, NAE, Royal Society.
15. David L. Dill. Donald E. Knuth Professor, Emeritus, Stanford. Fellow, IEEE, ACM. Member, NAE, AAoAS. Computer-Aided Verification Award, Alonzo Church Award.
16. Dawson Engler. Professor, Stanford. ACM Grace Murray Hopper Award, Mark Weiser Award, Numerous Best Paper awards.
17. Bob Frankston. Co-founder, Software Arts. Implemented VisiCalc (first spreadsheet). Fellow, IEEE, ACM, CHM. ACM Software System Award.

18. Neal Gafter. Software Architect, Facebook: Lead, Machine Learning Programming Language framework. Previously Principal Engineer, Microsoft: Technical lead, Roslyn Project; Software Engineer, Google; Senior Staff, Sun Microsystems. Developed C++, Java, and C# languages and compilers.
19. Erich Gamma. Microsoft Technical Fellow. Co-author, Design Patterns: elements of reusable object-oriented software, which won ACM Programming Language Award. Previously, Distinguished Engineer, IBM. ACM Software System Award.
20. Andrew Glover. Director, Delivery Engineering, Netflix. Steering Committee Chair, Spinnaker Open Source project. Author, Java Testing Patterns.
21. Allan Gottlieb. Professor, NYU. Led Ultracomputer group which introduced fetch-and-add instruction still in use today.
22. Robert Harper. Professor, Carnegie-Mellon. Co-designer, Standard ML programming language. Allen Newell Medal for Research Excellence, Herbert Simon Award for Teaching Excellence. Fellow, ACM.
23. Maurice Herlihy. Professor, Brown. Previously, Carnegie-Mellon. Dijkstra Prize in Distributed Computing, Gödel Prize in theoretical computer science, Fulbright Distinguished Chair. Fellow, ACM, AAoAS, National Academy of Inventors.

24. Tom Jennings. Faculty, Calarts Art+Technology Program, retired. Co-wrote Phoenix Software’s IBM compatible ROM BIOS. Creator of FidoNet, the first and most influential message and file networking system.
25. Alan Kay. Pioneer in object-oriented programming, personal computing, GUIs. Co-author, Smalltalk programming language. Positions at HP, Disney, Apple, Xerox PARC, Atari. ACM Turing Award, NAE Draper Prize, Kyoto Prize. Member, AAAS, NAE, AAoAS. Fellow, ACM, CHM, Royal Society of Arts.
26. Brian Kernighan. Professor, Princeton. Unix pioneer, Bell Labs. Co-creator, AWK programming language. Co-author, 13 books including seminal work on C programming language. Member, NAE, AAoAS.
27. David Klausner. Fifty years software/hardware experience at Microsoft, AT&T, Cisco, IBM, Hewlett Packard, Intel.
28. Kin Lane. Computer scientist working on API technology, business, politics. Twenty years' API experience as programmer, architect, executive, and currently the Director of Postman Open Technologies.
29. Ed Lazowska. Professor, University of Washington. Member, NAE, Washington State Academy of Sciences. Fellow, ACM, IEEE. Member, NAE, AAoAS. Past co-chair, President’s Information Technology Advisory Committee.

30. Douglas Lea. Professor and Department Chair, SUNY Oswego. Creator of Java concurrency APIs. Author, Concurrent Programming in Java. Dahl–Nygaard Prize. Fellow, ACM.
31. Bob Lee. CEO, Present Company. Previously, CTO, Square; Staff Engineer, Google. Led Android core library team, created Guice framework.
32. Harry Lewis. Professor, Harvard. Students included Bill Gates, Mark Zuckerberg. Previously dean, Harvard College; interim dean, Harvard's School of Engineering and Applied Sciences.
33. Douglas McIlroy. Professor, Dartmouth. Headed Bell Laboratories department that originated Unix. Many contributions to Unix including pipes abstraction. Designer, PL/I programming language. USENIX lifetime achievement award, programming tools award. Fellow, AAAS. Member, NAE.
34. Paul Menchini. CTO & CISO, North Carolina School of Science and Mathematics. Previously, HP, Intel, GE. Edited IEEE VHDL Standard. Developed first commercially successful VHDL compiler. IEEE Senior Life & Inaugural Golden Core member.
35. James H. Morris. Professor Emeritus, Carnegie-Mellon. Previously dean, department head; Professor, UC Berkeley; Principal Scientist and Research Fellow, Xerox PARC. Co-inventor, Knuth-Morris-Pratt algorithm. Fellow, ACM.

36. Peter Norvig. Google Director of Research. Previously directed Google's search algorithms group. Co-author, Artificial Intelligence: A Modern Approach. Fellow, AAAI, ACM, AAoAS.
37. Martin Odersky. Professor, EPFL (Lausanne, Switzerland). Creator, Scala programming language. Designed original Java generics. Wrote Java compiler.
38. David Patterson. Professor Emeritus, Berkeley. Previously Director, Parallel Computing Lab; Chair, CS Division; Chair, Computing Research Association; President, ACM. Projects included Reduced Instruction Set Computers (RISC), Redundant Arrays of Inexpensive Disks (RAID), and Network of Workstations. All led to multibillion-dollar industries. Forty honors including ACM Turing Award, IEEE John von Neumann Medal. Member, NAE, NAS, AAoAS. Fellow, AAAS, CHM, ACM, IEEE.
39. Tim Peierls. President, Seat Yourself. Previously, VP, Descartes Systems Group; MTS, Bell Labs. Member, four expert groups developing Java API specifications. Co-author, Java Concurrency in Practice.
40. Curtis Schroeder. Computer Scientist, Draper. Served as editor for widely reimplemented SISO CIGI API. Previously, Antycip Simulation, Lockheed Martin.

41. Robert Sedgewick. Founding chair and professor, Princeton CS Department. Co-inventor, Red-Black tree data structure. Author, 20 books including million-selling Algorithms. Steele Prize, ACM Karlstrom Award. Fellow, ACM.

42. Mary Shaw. Professor, Carnegie-Mellon. Specialist in software engineering. National Medal of Technology and Innovation, ACM SIGSOFT Outstanding Research Award, IEEE Distinguished Women in Software Engineering Award. Fellow ACM, IEEE, AAAS.

43. Alfred Z. Spector. Writer. Previously, CTO, Two Sigma; VP of Research, Google; CTO, IBM Software; VP, IBM Services and Software; Professor, Carnegie-Mellon. Fellow, IEEE, ACM. Member, NAE, AAoAS. IEEE Kanai Award for Distributed Computing, ACM Software Systems Award.

44. Michael Stonebraker. Data base pioneer. Main architect, INGRES relational DBMS, POSTGRES object-relational DBMS. CTO, Paradigm4, Tamr; Professor, MIT. Previously Professor, UC Berkeley. ACM Turing Award, IEEE John von Neumann Medal, ACM System Software Award, SIGMOD Innovations Award. Member, NAE.

45. Ivan E. Sutherland. Professor, founder of Asynchronous Research Center, Portland State. Previously, Technical Fellow, Sun Microsystems. 1963 MIT Ph.D., Sketchpad, is widely known; he has been called "the father of computer graphics." ACM Turing Award, IEEE John von Neumann Medal, Kyoto Prize. Fellow, ACM, CHM. Member, NAE, NAS.
46. Andrew Tanenbaum. Professor emeritus, Vrije Universiteit. Principal designer, Linux-precursor MINIX. Author, 24 books . Member, Royal Netherlands Academy of Arts and Sciences. Fellow ACM, IEEE. USENIX Lifetime Achievement Award, Eurosys Lifetime Achievement Award.
47. Brad Templeton. Founder, ClariNet (perhaps the earliest dot-com company). First employee, Personal Software/Visicorp (first major microcomputer applications company). Author, numerous microcomputer software titles. Chairman Emeritus, EFF.
48. Andries van Dam. Professor, Brown University. Cofounder ACM SICGRAPH. Co-author Computer Graphics: Principles and Practice. Fellow IEEE, ACM. Member NAE, AAoAS. Numerous awards including IEEE Centennial Medal.
49. Guido van Rossum. Created Python programming language. Was Principal Engineer, Dropbox; Senior Staff, Google. ACM Distinguished Engineer. Fellow, CHM, CWI Dijkstra.

50. John Villasenor. UCLA professor of electrical engineering, law, and public policy. Director of the UCLA Institute for Technology, Law, and Policy. Brookings Institution senior fellow. Hoover Institution senior fellow.
51. Jan Vitek. Professor, Northeastern. Specialist in programming languages. Chief Scientist, Fiji Systems. Past Chair, ACM SIGPLAN.
52. James Waldo. Professor, CTO, Harvard. Was Distinguished Engineer, Sun Microsystems; developed Java APIs for distributed systems. Author, Java: The Good Parts.
53. Dan Wallach. Professor, Rice University. Rice Scholar, Baker Institute for Public Policy. Former member, Air Force Science Advisory Board, USENIX Board of Directors.
54. Frank Yellin. Original member, Sun Microsystems' Java Project. Co-author, The Java Virtual Machine Specification, Java API specification. Formerly Google, Lucid.

UNITED STATES COURT OF APPEALS FOR THE FEDERAL CIRCUIT

CERTIFICATE OF COMPLIANCE WITH TYPE-VOLUME LIMITATIONS

Case Number: 2021-1542

Short Case Caption: SAS Institute, Inc. v. World Programming Limited

Instructions: When computing a word, line, or page count, you may exclude any items listed as exempted under Fed. R. App. P. 5(c), Fed. R. App. P. 21(d), Fed. R. App. P. 27(d)(2), Fed. R. App. P. 32(f), or Fed. Cir. R. 32(b)(2).

The foregoing filing complies with the relevant type-volume limitation of the Federal Rules of Appellate Procedure and Federal Circuit Rules because it meets one of the following:

- [✓] the filing has been prepared using a proportionally-spaced typeface and includes 6005 words.
- [ ] the filing has been prepared using a monospaced typeface and includes __________ lines of text.
- [ ] the filing contains __________ pages / __________ words / __________ lines of text, which does not exceed the maximum authorized by this court's order (ECF No. __________).

Date: 08/30/2021

Signature: /s/Jeffrey Theodore Pearlman

Name: Jeffrey Theodore Pearlman