No. 18-956

IN THE

Supreme Court of the United States

GOOGLE LLC,

Petitioner

v.

ORACLE AMERICA, INC.,

Respondent

ON WRIT OF CERTIORARI TO THE UNITED STATES COURT OF APPEALS FOR THE FEDERAL CIRCUIT

BRIEF OF *AMICUS CURIAE* INTERDISCIPLINARY RESEARCH TEAM ON PROGRAMMER CREATIVITY IN SUPPORT OF RESPONDENT

Ralph D. Clifford
Counsel of Record for *Amicus Curiae*
University of Massachusetts School of Law
333 Faunce Corner Road
North Dartmouth, MA 02747
rclifford@umassd.edu
(508) 985-1137

Dated: February 13, 2020

BATEMAN & SLADE, INC. STONEHAM, MASSACHUSETTS

Table of Contents

Table of Cited Authorities

Interest of the *Amicus Curiae*[1]

The *amicus* is comprised of five individuals who are an interdisciplinary team that is researching programmer creativity ("Research Team"). Each member is a professor with the expertise described below:

- Ralph D. Clifford, the principal investigator, is a professor of law at the University of Massachusetts School of Law who specializes in intellectual property and cyberlaw issues, specifically including the requisite creativity needed for copyright. *See* Ralph D. Clifford, *Intellectual Property in the Era of the Creative Computer Program: Will the True Creator Please Stand Up?*, 71 Tul. L. Rev. 1675 (1997); Ralph D. Clifford, *Random Numbers, Chaos Theory and Cogitation: A Search for the Minimal Creativity Standard in Copyright Law*, 82 Denver L. Rev. 259 (2004) [hereinafter Clifford, *Random Numbers*]; Ralph D. Clifford, *Creativity Revisited*, 59 IDEA 25 (2018). Before obtaining his law degree, he studied computer science at the undergraduate

[1] This brief was written exclusively by the *amicus's* counsel with the exclusive generous financial support of the University of Massachusetts—Dartmouth. No counsel for a party authored any portion of this brief. The institutional affiliation of the individuals comprising the *Amicus* are for identification only and do not represent the position of their institutions.

This brief is submitted with the blanket consent of Google LLC and the written consent of Oracle America, Inc.

level and practiced computer programming professionally for ten years.

- Firas Khatib is an Assistant Professor of Computer and Information Science at the University of Massachusetts—Dartmouth who specializes in bioinformatics and citizen science. *See* Firas Khatib et al., *Players, Algorithm Discovery by Protein Folding Game Players*, Proc. of the Nat'l Acad. of Sci. U.S.A. (2011), https://doi.org /10.1073/ pnas.1115898108.

- Trina Kershaw is an Associate Professor of Psychology at the University of Massachusetts—Dartmouth who specializes in cognitive processes, the creative process, and creativity measurement in laboratory settings and in engineering design. *See* Trina Kershaw et al., *A Decision Tree Based Methodology for Evaluating Creativity in Engineering Design,* Frontiers in Psychology (2019), https://doi.org/ 10.3389/fpsyg.2019.00032.

- Kavitha Chandra is the Associate Dean and Professor of Electrical and Computer Engineering at the Francis College of Engineering, University of Massachusetts—Lowell who specializes in computational acoustics and creativity in engineering.

- Jay McCarthy is an Associate Professor of Computer Science, University of Massachusetts—Lowell who specializes in the analysis of computer programs and programming languages, especially for the purposes of verifying the correctness and equivalence of different programs that attempt to do the same thing. *See* Jay McCarthy, *A Programmable Programming Language*, Comm. of the ACM (Mar. 2018); Jay McCarthy, *Model Checking Task Parallel Programs for Data-Race*, NASA Formal Methods Symp. (2018); Jay McCarthy, *Cryptographic Protocol Explication and End-Point Projection*, European Symp. on Research in Comp. Sec. (2018).

Summary of Argument

This brief answers the two primary issues that are associated with the first question before the Court. First, the programmers' expression of the Java-based application programmer interfaces ("APIs") are sufficiently creative to satisfy that requirement of copyright law. Second, the idea-expression limitation codified in Section 102(b) of Copyright Act does not establish that the APIs are ideas. Both of these assertions are supported by the empirical research undertaken by the Research Team.

This brief expresses no opinion on the resolution of the fair use question that is also before the Court.

I. Creativity

Feist Publications, Inc. v. Rural Tel. Serv. Co., 499 U.S. 340 (1991), teaches that all works must be the result of creative expression in order to qualify for a copyright. However, as *Feist* specifically addressed the white pages of the classic telephone book, *id.* at 342, little guidance is provided for dealing with more expressive works such as computer programs. Further, language in *Feist* instructs that copyrighted works should be based on "creative spark." *Id.* at 345. Unfortunately, this direction does little to explain how the "spark" is to be identified, a problem that is compounded by the dual expressive-functional characteristics of a computer program.

As a practical matter, without turning *Feist's* creativity requirement into a subjective analysis of how the particular author functioned during the work's creation, or allowing it to become an excuse for the judicial censorship much feared by this Court more than a hundred years ago in *Bleistein v. Donaldson Lithographing Co.*, 188 U.S. 239, 251–52 (1903).[2] an objective measurement is needed. Consequently, "creativity" should be found where it is apparent that the author had many different ways a particular idea could have been expressed, from which the author made an intellectually-based selection. *See* Clifford, *Random Numbers* at 295–96.

[2] "It would be a dangerous undertaking for persons trained only to the law to constitute themselves final judges of the worth of pictorial illustrations, outside of the narrowest and most obvious limits. ... At the other end, copyright would be denied to pictures which appealed to a public less educated than the judge."

When this is applied to the APIs in question in this litigation ("Oracle's APIs"), it is clear that the *Feist* creativity requirement is met. *See Oracle Am., Inc. v. Google Inc.*, 750 F.3d 1339, 1354 & 1356–57 (Fed. Cir. 2014) (case below). As our research demonstrates, even the simplest computer program is capable of being expressed in many ways.[3] As programs become more complex, the number of unique solutions also increases.[4] Consequently, as there are clearly choices for a programmer to make from a wide variety of expressions, *Feist* creativity exists for the vast majority of computer programs including Oracle's APIs.

II. Idea/Expression Dichotomy

Section 102(b) of the Copyright Act codifies the requirement that a copyright's protection be limited to the expression created by the author, expressly excluding the ideas that underlie the creation. Separating ideas from expressions has never been easy; indeed, courts have long struggled with establishing guidance on how to make this distinction. *See, e.g., Nichols v. Universal Picture Corp.*, 45 F.2d 119, 121 (2d Cir. 1930) (L. Hand, J.). When applied to the technical writing that is programming, drawing the distinction becomes more

[3] The Research Team's preliminary study involves correctly functioning code submitted by multiple programmers to solve the same problem. The simplest program in our research set—searching for the most frequent character pattern within a larger string—demonstrated a large variety of solutions: 20 unique solutions were created by the 27 programmers.

[4] On the most complex program in the research set, 23 unique solutions were submitted by the 26 programmers who solved the problem.

difficult as those defining the line rarely have sufficient technological background to inherently understand what the programmer has written.

Despite the difficulty of analysis, there is a developing accord among the circuit courts that the analytical approach established for literary works in *Nichols* is an appropriate approach for separating the ideas and expressions within a computer program. *See, e.g., Computer Assocs. Int'l, Inc. v. Altai, Inc.*, 982 F.2d 693, 706 (2d Cir. 1992); *Apple Computer, Inc. v. Microsoft Corp.*, 35 F.3d 1435, 1445–46 (9th Cir. 1994). *But see Whelan Assoc., Inc. v. Jaslow Dental Laboratory, Inc.*, 797 F.2d 1222, 1232 (3d Cir. 1986). This analytic approach was adopted by the Federal Circuit in the case at bar, *see Oracle*, 750 F.3d at 1357 (applying 9th Circuit precedents), and should now be established as the appropriate methodology for all copyright cases, including those that involve computer software.

Care and guidance is needed, however, for the lower courts to properly apply the abstracting and filtering process to computer programs. First, as matters are abstracted, care is needed in distinguishing between public domain ideas and public domain expressions. Reusing a public domain *idea* does not impact the expression's copyrightability as all are free to re-express the idea. *See Bleistein,* 188 U.S. at 249.[5] Only if the author is attempting to recapture a public domain *expression* should the courts prevent the attempt. Second, artificial distinctions should not be imposed on computer

[5] "Others are free to copy the original. They are not free to copy the copy."

programs because they are assertedly the result of engineering-based rather than artistically-based human inspiration. As our research demonstrates, *expressive* creativity underlies both types of inspiration at a level sufficient to satisfy *Feist.* The conclusion that directly flows from this is that, as with other literary works protected by copyrights, computer programs are primarily expressive.

The secondary conclusion that flows from this is the limited applicability of the merger doctrine in evaluating computer software copyrights. Rather than only having a limited number of expressions available, the programmer has a large number from which to choose.

Argument

I. The Court Should Clarify the Definition of the "Creativity" Needed Under *Feist* to Provide an Objective Test Based on the Author Having Had Available Multiple Variations of Expression from Which an Intellectual Choice Was Made

In *Feist Publications, Inc. v. Rural Telephone Service Co.,* 499 U.S. 340 (1991), this Court established that there is a threshold of creativity that must be contained within a compilation of facts in order for a copyright to subsist. *See id.* at 348. The Court stated:

> [T]he work [must be] independently created by the author (as opposed to copied from other works), and [must] possess[] at least some minimal degree of creativity. To be sure, the requisite

> level of creativity is extremely low; even a slight amount will suffice. The vast majority of works make the grade quite easily, as they possess some creative spark, no matter how crude, humble or obvious it might be. Originality does not signify novelty; a work may be original even though it closely resembles other works so long as the similarity is fortuitous, not the result of copying.

Id. at 345 (citations and quotation marks omitted).

While the Court's *Feist* opinion suggests that this creativity requirement is a requisite of all copyrighted works, not just compilations, *see id.*, the Court's discussion of creativity in the context of a factual compilation provides clouded guidance as to the nature of the requirement for more fanciful works such as the computer programs in the case at bar. The resulting contradictory holdings of the circuit courts when addressing fanciful works bear witness to the need for a clarifying ruling on the meaning of copyright creativity. *Compare, e.g., Satava v. Lowry*, 323 F.3d 805 (9th Cir. 2003) (disallowing copyright in an artistic glass jellyfish sculpture) *with Boisson v. Banian, Ltd.*, 273 F.3d 262 (2d Cir. 2001) (allowing copyright in much simpler quilt design). This same confusion concerning the appropriate standard arises in the evaluation of computer programs as stated by the court below: "Circuit courts have struggled with, and disagree over, the tests to be employed when attempting to draw the line between what is protectable expression and what is not." *Oracle*, 750 F.3d at 1357. Consequently, this Court should clarify the standard of creativity for works such as those at bar that are more fanciful than the white pages from a telephone book.

At its core, *Feist* seemed most concerned with the choices that were available to and made by the author. *See Feist*, 499 U.S. at 348. The Court stated,

> The compilation author typically chooses which facts to include, in what order to place them, and how to arrange the collected data so that they may be used effectively by readers. These choices as to selection and arrangement, so long as they are made independently by the compiler and entail a minimal degree of creativity, are sufficiently original that Congress may protect such compilations through the copyright laws.

Id.

Producing computer programs and other more fanciful works clearly exceeds the mere "choosing" and "ordering" found in a factual compilation. To clarify how the *Feist* holding applies to these more fanciful works, the two prerequisites required of a compilation should be restated and required of any work, including Oracle's APIs. Authors of fanciful works do not choose and order facts; instead, they choose how to express a concept from the multitude of ways in which this can be done. This choice-making can serve as the foundation for an objective creativity test for non-compilation works. In other words, to be copyrightable, all works must result from their authors making choices from a multitude of possible ways of expressing the work. *See* Clifford, *Random Numbers* at 295–96.

Determining if this has occurred would be practical as part of the fact-finding of the courts. To determine if the minimum creativity required is present, the court would need to examine the work to determine that the author had choices and made a *decision* to express the work in the way it was done. Fundamentally, to determine if creativity exists within a work, it must be established that the author "ma[d]e a judgment, ... determine[d] a preference; [or] c[a]me to a conclusion," *Decide, Random House Unabridged Dictionary* 517 (2d Ed. 1993), about the expression used. Only where this decision making is apparent can the courts be sure that the "modicum of creativity" required by *Feist*, 499 U.S. at 362, exists within the work.

When this type of test is applied to computer software, our research shows that the typical program complies. In our initial research protocol, 27 student programmers were given four problems to solve that required them to write software.[6] Each submitted program was tested and successfully solved the assigned problem before it was included in the research dataset.

Having built the dataset, the code generated was analyzed based on the number of each "fundamental programming construct" the

[6] The students were all in a course that addressed using computer technology to solve processing problems associated with DNA research.

programmer had used to produce the code.[7] By determining all of the constructs used by each programmer, expressive differences in the code was captured based on the different choices of constructs made by the programmers. As an example, one programmer might have chosen to write part of the code based on a "for" loop while another might have chosen a "while" loop. Ultimately, as both programs achieved a solution to the same problem, the choice of which loop type to use becomes expressive as neither has computational advantage over the other. By accumulating all of these differences over all of the different types of constructs, an overall program description code could be created. If two programs had the same description code assigned to it, they were expressively the same;[8] if the description code differed, significantly different ways were used to express the same programming function.

The first problem assigned to the programmers was to determine the most frequent character pattern of a certain size within a larger string. This problem is not trivial to code but is also not computationally complex. Most programmers would be able to correctly code a solution within a few hours. When the

[7] A "programming construct" is a particular instruction that all programming languages provide. Our research identified seven of these: subroutines, for loops, while loops, if statements, else statements, case statements, and go to statements.

[8] This excludes differences based on the variables and other names chosen by the programmer. Our research also captures these differences, but has discounted them here as a change in variable name, standing alone, is the kind of trivial variation that is given little credence in copyright law. *Cf. Downing v. Abercrombie & Fitch*, 265 F.3d 994, 1004 (9th Cir. 2001) ("A person's name ... is not a work of authorship....").

programs solving this problem within our dataset were analyzed, however, the number of unique solutions submitted was significantly large. In fact, there were 20 unique solutions among the 27 programs written to solve the problem, a percentage of variation of 74.1%.

As would be expected, more complex problems had a larger number of unique solutions. A later assignment given to the programmers required that they develop code that solves for what is known as a "greedy motif search with pseudocounts" problem.[9] Twenty-three unique solutions were submitted among the 26 solutions submitted (one programmer failed to submit a valid program), a percentage of variation of 88.5%.

What this demonstrates is that there are a multitude of programs that can be expressed to solve even fairly trivial computer programming problems. More directly, there are a large number of expressive choices from which each programmer-author can

[9] The goal of the "greedy motif search" algorithm is to find similar motifs in long segments of DNA sequences. A "motif" is a short string of DNA that denotes the location in the full DNA string where a regulatory protein should attach in order for the DNA to carry out its gene expression purpose.

The search algorithm is complicated because motifs from similar DNA sequences contain minor variations and are therefore not identical. This introduces the need for probability determinations when comparing a potential motif to a DNA string. As probabilities of zero would cause significant processing problems, "pseudocounts" are used to mathematically prevent zeros from occurring.

choose an expression of his or her desire.[10] *Feist's* standard of creativity based on making choices among expressions has been established.

Of course, in the case at bar, the degree of complexity of the software in litigation far exceeds the relatively uncomplicated programs in our research dataset. Oracle's APIs involve thousands of lines of code to define the "overall system of organized names—covering 37 packages, with over six hundred classes, with over six thousand methods." *Oracle*, 750 F.3d at 1351. In creating Oracle's APIs, the programmers created at least "thousands of individual elements" resulting in "7,000 lines of declaring code" as that was what Google copied. *Id.* at 1349 & 1353. Within these thousands of methods and lines of code, numerous expressive decisions were made. As a de minimus example, even the choice to call the example function described by the court below "MAX" rather than "MAXIMUM" or "LARGER," represents an expressive choice. *See id.* at 1349–50.

Consequently, the Court should find that sufficient creativity exists in Oracle's APIs to satisfy the *Feist* creativity standard. This requires an examination of the idea/expression dichotomy.

[10] If the programmer's choice of variable names is included, every program becomes completely unique.

II. The Court Should Interpret the Idea/Expression Dichotomy, 17 U.S.C. § 102(b), so that the Expressive Nature of Computer Software Remains Protected by Copyright

Distinguishing between an idea and its expression has never been easy. *See Nichols v. Universal Picture Corp.*, 45 F.2d 119, 121 (2d Cir. 1930) (L. Hand, J.). *See also Holmes v. Hurst*, 174 U.S. 82, 86 (1899). Despite the analytical difficulty required, the fundamental approach established by Judge Hand—abstracting the content of the work at decreasing levels of detail and searching among these abstractions for the point where allowing the copyright would result in the copyright preempting the underlying idea, *see Nichols*, 45 F.2d at 121—provides a compelling and logical solution to the problem of applying Section 102(b) presented by the case at bar. Indeed, this abstraction and examination approach has been widely adopted and is part of the most widely accepted test among the Circuit Courts of Appeal for copyright infringement of a computer program, *see, e.g., Computer Assocs. Int'l, Inc. v. Altai, Inc.*, 982 F.2d 693 (2d Cir. 1992); *Gates Rubber Co. v. Bando Chem. Indus., Ltd.*, 9 F.3d 823, 834 (10th Cir. 1993); *Bateman v. Mnemonics*, 79 F.3d 1532, 1543–46 (11th Cir. 1996); *Oracle*, 750 F.3d at 1355–56 (case below). This analytical approach has not been universally adopted, however, as the Third Circuit has endorsed a much more intuitive approach. *See Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc.*, 797 F.2d 1222 (3d Cir. 1986).

A. The Court Should Adopt the Abstraction-Filtration-Comparison Test to Separate Ideas from Expressions in a Computer Program

Upon examination, the *Computer Assocs.* approach—the Abstraction-Filtration-Comparison Test—is more meritorious than *Whelan*'s intuitive approach.[11] It is vital that courts scrutinize computer programs that are claimed for copyright to enforce the idea/expression dichotomy found in Section 102(b). As with other fact-based works, the intertwining of expressions and ideas within a computer program require a critical examination and dissection of it. This is made more crucial as computer software both expresses and implements the programmer's code. Further, unlike English language works, computer programs are not communicative to non-specialists, limiting the power of an intuitive approach to reach a valid conclusion.

Consequently, this Court should adopt the Abstraction-Filtration-Comparison Test to implement section 102(b). This approach, described in *Computer Assocs.*, 982 F.2d at 706–11, provides the appropriate foundation for understanding the dividing line between ideas and expressions in computer programs. Unlike the *Whelan* approach, this test requires a careful consideration of the copyrighted software and insures that its author

[11] *Whelan* is mostly criticized here for its approach and its limits. The *Whelan* court quite accurately excluded the basic business purpose of the software in the case under 17 U.S.C. § 102(b), but failed to examine the code for other ideas that should have also been excluded from the protections of copyright law.

maintains rights to the expression while allowing all others use of any contained ideas. *Whelan* does not do this. Limiting the idea in computer software to what the overall purpose of the program is designed to achieve, *see Whelan*, 797 F.2d at 1236, removes many ideas (both of business processing and computer programming) from use by improperly including these within the copyright.

B. Guidance is Needed from the Court on What Matters are Properly Filtered out of the Expression Within a Computer Program

1. Proper Filtering Recognizes the Expressive Nature of Computer Programs

Computer Assocs. was mostly on point about the details of how to exclude ideas from coverage by the copyright, expressed in the "filtering" part of the court's tripartite test. *See Computer Assocs.*, 982 F.2d at 707–10. When a particular expression is present within software only because that expression is needed in order for the software to operate on the target hardware, *Computer Assocs.* was correct in excluding that statement from consideration as part of the expression. *See id.* at 709–10. Indeed, our research indirectly confirms this as the machine-oriented programming constructs were excluded from our dataset as including them mis-characterized the similarities within the different programmer's code. Similarly, the presence or absence of any given fundamental programming construct in itself should not be considered expression as these are required to produce all computer programs written in procedural languages. As the line moves from the individual statement types to selected combinations of the constructs, however, the copyright line between idea and expression has been crossed and the realm of expressions has been reached. As our research demonstrates, even simple programming tasks can result in a wide range of possible expressions, negating an assertion that computer programs are mostly ideas rather than the expression of them. Instead, our research establishes that computer programs are highly expressive with significant variations existing in how even the simplest program is written. Our research has established this in two ways.

First, we examined the number of unique versions of each program that were submitted.[12] These calculations showed that almost 75% of the simplest programs were different from all of the others and almost 90% of the more complicated programs varied. Based on this, we determined that variation was the norm, not the exception.

Second, to confirm our preliminary findings, the Research Team subjected the multiple versions of the four programs to formal statistical analysis. By examining the average number of times each fundamental programming construct was used in comparison to each's standard deviation, the large degree of variation was clear. Surprisingly, for three of the constructs (subroutines, for loops, and else statements), the standard deviation was actually larger than the average. While the inherent meaning of this is limited, it does suggest that the data points

[12] The methodology used to make this determination is set forth above. *See supra* pp. 10–13.

are widely scattered and may exist without a defining pattern. In other words, the programmers do not choose to use the constructs based on any defined underlying rule; rather, they are making intellectually-based choices among the possible expressions. To test this, we assumed the opposite and performed an univariate ANOVA (analysis of variance) based on the constructs each example program used. An ANOVA procedure is a way to test if there are significant mean group differences on a variable of interest. *See Sarah Boslaugh & Paul Watters, Statistics in a Nutshell* 232–38 (2008). For example, an ANOVA can determine if the number of "for" loops used by each programmer was compelled by a factor such as the algorithm rather than individual choice. In other words, a non-significant ANOVA test would establish that there was no expressive creativity used in the programming effort.

In fact, our analysis established a high degree of variation among the choices made by the programmers with F-test scores ranging from 11.97 through 39.51.[13] Values this high on an F-test is consistent with a large degree of variation in the code and rejects the null hypothesis that there is an underlying common justification for the choice of which set of programming constructs to use. To summarize the statistical analysis, it shows that the choice of programming constructs by each programmer are unconstrained by a common variable. Consequently, programmers are making creative expressive choices.

[13] The p-test score for these results were less than .0001, far smaller than the minimum (.05) required for statistical significance.

Because of this kind of difference being found within computer code, it is unlikely that the copyright merger concept, *see Herbert Rosenthal Jewelry Corp. v. Kalpakian*, 446 F.2d 738 (9th Cir. 1971), will provide any help for evaluating computer software. As a result, the district court's reliance on the merger doctrine to invalidate the Oracle copyright, *see Oracle Am., Inc. v. Google Inc.*, 872 F. Supp. 2d 974, 998 (N.D. Cal. 2012) (case below), was misplaced and the Federal Circuit was correct to overrule the lower court's decision on this ground. *Oracle*, 750 F.3d at 1360. Our research has established that even for the simplest code in the dataset (programs that are 25 to 50 lines in length), so many alternative methods of expression existed that asserting that the idea and expression have merged in any computer program—except the most trivial—is unsustainable. When the degree of coding variation found within computer code is scaled up to software on the scope of Oracle's APIs, merger is impossible.

The fact that the copyrighted expression in litigation is only the declaring code rather than the operational or implementing code, *see Oracle*, 750 F.3d at 1349, does not change this analysis. While there were obviously considerably more choices made by the programmers who created the operable aspects of Oracle's APIs, more than enough creative choices were made by them in creating the 7,000 lines of declaring code, *see id.* at 1349 & 1353, to satisfy the copyright requirements.

In summary, when Oracle's APIs are examined, it is clear that there were thousands of different ways the APIs could have been written when they were created. As a result, the Federal Circuit was correct to determine that they are protected from copying by the Copyright Act.

2. Proper Filtering Does Not Establish Improper Barriers to Protecting Programming Expressions

Although our research established the viability of the abstraction-filtration-comparison test, the *Computer Assocs.* court made a significant mistake when it attempted to address the overall efficiency of a computer algorithm. According to the Second Circuit, the more efficient a computer program is, the more likely it is to be an idea rather than an expression. *See Computer Assocs.*, 982 F.2d at 707–09 (treating efficiency as an idea as a matter of law). This holding is based on a mistaken view of efficiency both within computer programming and within the broader engineering disciplines in which it resides and is inconsistent with our research.

The *Computer Assocs.* court misunderstood efficiency as it exists within an engineering discipline such as computer science. Unlike the court's view, there is no single point of efficiency that programs attempt to reach with success meaning that an idea has been reached. Instead, all programming efforts result in the programmer balancing a multitude of considerations that are often contradictory.[14] In

[14] This same thing is true in all other engineering disciplines. Recently, for example, the Tappan Zee bridge across the Hudson River in New York was replaced. *See* Karen DeWitt, *Tappan Zee Bridge Set to Open*, https://www.northcountrypublicradio.org/news/story/34544/20170824/tappan-zee-bridge-

programming, for example, greater speed can often be achieved, but only as an escalating cost. Even speed of operation, itself, may not be the most important criteria; indeed, in earlier programming efforts, limiting the amount of storage space that was needed by the program was typically far more critical than achieving blinding processing speed.[15] Furthermore, when applied to something as complex as computer software, determining efficiency becomes extraordinarily difficult as most times the evaluation has to be reduced to probabilities as the data being processed can radically affect the resulting speed. *See* Donald E. Knuth, *Fundamental Algorithms* § 1.2.10 (2d ed. 1975). In summary, achieving some type of efficiency within a computer program does not transform it into an idea.

Similarly, the courts below have often failed to carefully distinguish between public domain *expressions* which should be filtered out of a work and public domain *ideas* which should not. *See, e.g., Computer Assocs.*, 982 F.2d at 714–15; *Satava*, 323

---

set-to-open-questions-remain-on-future-tolls. Determining which of the two bridges is the more "efficient" one makes no real sense. Does that analysis focus on cost? The number of cars that can be carried? The size of the largest truck that can safely cross? The bridge's attractiveness? Clearly, "efficiency" is not a single point.

[15] This need for storage efficiency lead to a problem that was known as the "Y2K" problem as programmers, for years, had saved storage space by not storing the "19" that was associated with the year; instead,1965 would be stored as 65, thus saving ½ of the needed space. *See* Josh Hodas, *What are the main problems with the Y2K computer crisis and how are people trying to solve them?*, Sci. Am., June, 1999, https://www.scientificamerican.com/article/what-are-the-main-problem.

F.3d at 810. As this Court best expressed it in *Bleistein,* 188 U.S. at 249, "[o]thers are free to copy the original. They are not free to copy the copy." In the case at bar, Google was free to re-express the idea of APIs in general, including the functional characteristics needed by application programmers.[16] Google was not free merely to copy Oracle's expression.

Finally, as the Federal Circuit ruled below, Google's desire to achieve interoperability—or more accurately, its goal to reduce the chance that its programmers would be confused by a different API system—is not relevant to whether Oracle's APIs were ideas. *See Oracle*, 750 F.3d at 1371.[17] Ideas exist independently of how another may wish to use it. By analogy, many may wish to use the Hogwarts world created by J.K. Rowling or the Middle-Earth world created by J.R.R. Tolkien, but both are creative expressions of their authors and, thus, are protectable by copyright. Of course, neither author's copyright protects the idea of having a world of magic, so future

[16] Indeed, the general concept of an API has existed by that name since at least the early 2000s. *See* Mary Sweeney, *Visual Basic for Testers* 211 (2001) (discussing the "APIs" used in Microsoft Windows). Of course, the concept without the name existed for decades before that. *See Macro Assemblers*, *Encyclopedia of Comp. Sci.* 99–100 (Anthony Ralston et al. ed. 4th ed. 2000) (describing achieving standard programming tasks by using the macro system available with 1960–1980-era IBM computers); IBM Corp., *OS/VS-VM/370 Assembler Programmer's Guide* 69 (5th ed. 1982) (defining "library macro definition" as "IBM-supplied ... macro definitions").

[17] As the Federal Circuit acknowledged, interoperability may be relevant to a fair use analysis. *See id.* at 1369–70. This brief takes no position on this question.

authors can always express their own versions. Similarly in the case at bar, Oracle was free to express its own "world" of APIs. If Google wants one too, it is free to create one. Google should not be free, however, to appropriate what Oracle had already expressed.

Conclusion

For the reasons discussed above, the Court should affirm the decision of the Federal Circuit that the Respondent's software APIs are protected by valid copyrights. The APIs are creative expressions worthy of copyright protection. Providing this protection will not stop others, including Google, from developing its own set of APIs.

Respectfully submitted,

Interdisciplinary Research Team on
Programmer Creativity

Ralph D. Clifford
Counsel of Record for *Amicus Curiae*
University of Massachusetts School of Law
333 Faunce Corner Road
North Dartmouth, MA 02747
rclifford@umassd.edu
(508) 985-1137

Dated: February 13, 2020