CINDY A. COHN, ESQ.; SBN 145997
McGLASHAN & SARRAIL
Professional Corporation
177 Bovet Road, Sixth Floor
San Mateo, CA 94402
Tel: (415) 341-2585
Fax: (415) 341-1395

LEE TIEN, ESQ.; SBN 148216
1452 Curtis Street
Berkeley, CA 94702
Tel: (510) 525-0817

Attorneys for Plaintiff
Daniel J. Bernstein

IN THE UNITED STATES DISTRICT COURT

FOR THE NORTHERN DISTRICT OF CALIFORNIA

DANIEL J. BERNSTEIN)
) C 95-00582 MHP
Plaintiff,)
) DECLARATION OF
v.) HAROLD ABELSON
))
UNITED STATES DEPARTMENT OF)
STATE et al.,))
Defendants.)
)
)

I, HAROLD ABELSON, hereby declare:

1. I am Professor of Computer Science and Engineering at the Massachusetts Institute of Technology in Cambridge, Massachusetts. I give this declaration in my personal capacity and not on behalf of MIT.
2. I have been active in computer science at MIT since 1969, when I began as a graduate student. I joined the MIT Department of Electrical Engineering and Computer Science as a faculty member in 1977.
3. Since 1981, I have been charge of MIT's introductory computer science subject, "Structure and Interpretation of Computer Programs" (together with my colleague, Prof. Gerald Jay Sussman). This subject is taught at MIT each year to between 500 and 700 undergraduates.
4. Together with Gerald Jay Sussman and Julie Sussman, I am author of the textbook "Structure and Interpretation of Computer Programs," which is based on our MIT course. The first edition of this book was published by the MIT Press and the McGraw-Hill Book Company in 1985; the second edition will appear later this year.
5. Through the MIT course and the textbook, my work in computer science education has gained major visibility. Materials arising from our course are currently used at over 200 colleges and universities worldwide.
6. In recognition of my educational activities, I was named winner of 1995 Taylor L. Booth Award, given by the Institute of Electronic and Electrical Engineers (IEEE) Computer Society for outstanding contributions to computer science and engineering education. I was cited for continued contributions to the pedagogy and teaching of introductory computer science.

7. I comment from the perspective of someone who has been active in the teaching of university computer science.

COMPUTER PROGRAMS ARE A MEDIUM OF EXPRESSION

8. The notion that computer programs are a medium of expression is widespread throughout computer science education. In particular, this notion is central to the approach to computer science used at MIT over the past fifteen years. The idea appears prominently in the preface to our textbook:

Our design of this introductory computer-science subject reflects two major concerns. First, we want to establish the idea that a computer language is not just a way of getting a computer to perform operations but rather that it is a novel formal medium for expressing ideas about methodology. Thus, programs must be written for people to read, and only incidentally for machines to execute....

COMPUTER PROGRAMS EXPRESS IDEAS ABOUT METHODOLOGY

9. Just as ordinary mathematics and logic serve as a languages for expressing ideas about truth and falsehood (so-called "declarative knowledge"), computer programs serve as languages for expressing ideas about how to do things (so-called "imperative knowledge"). The following excerpt from a paper I wrote in 1990 expands on this point (from "Computation as a Framework for Engineering Education", in Research Directions in Computer Science: An MIT Perspective, Cambridge, MA: MIT Press, 1991):

10. "To illustrate the difference between declarative and imperative knowledge, consider the following definition of a square root:

The square root of a number X is the number Y such that Y times Y equals X.

This is declarative knowledge. It tells us something that is true about square roots. But it doesn't tell us how to find a square root.

11. "In contrast, consider the following ancient algorithm, attributed to Heron of Alexandria, for approximating square roots.

To approximate the square root of a positive number X

- Make a guess for the square root of X.
- Compute an improved guess as the average of the guess and X divided by the guess.
- Keep improving the guess until it is good enough.

12. "Heron's method doesn't say anything about what square roots are, but it does say how to approximate them. It is a piece of imperative "how to" knowledge.

13. "Computer Science is in the business of formalizing imperative knowledge -- developing formal notations and ways to reason and talk about methodology. Here is Heron's method formalized as a procedure in the notation of the Lisp computer language:

```
(define (sqrt x)
(define (good-enough? guess)
(<(abs (- (square guess) x)) tolerance))
(define (improve guess)
(average guess (/ x guess)))
(define (try guess)
(if (good-enough? guess)
guess
(try (improve guess))))
(try 1))
```

14. "Certainly, if the only things we ever computed were square roots, then Computer Science would not be of much interest. Similarly, if all one ever did in geometry was surveying, then geometry would not be of much interest. In each case, the importance of having a formalism is that it provides a framework for controlling complexity, a way to think about ideas that are too involved to think about all at once. The important techniques in Computer Science are the techniques for coping with methodological complexity."

EXPRESSIVE STYLE IS AN IMPORTANT ASPECT OF COMPUTER PROGRAMS

15. One indication that computer programs have major expressive elements is that teachers of programming regularly evaluate student programs on stylistic issues such as readability (by people) and appropriateness of choice of elements. This is analogous to the way that teachers of writing evaluate student essays on the style

and quality of writing, not just on the meanings of the words.
16. The criteria used in evaluating "programming style" are often unrelated to the sequence of operations that would be carried out by a computer in executing the program. Two programs might evoke exactly the same process when executed by a computer, and yet be judged very differently, because they express the process in different ways.
17. As an example, the following excerpt from our textbook (2nd edition) discusses a technique called "mapping over a list":
MAP is an important construct, not only because it captures a common pattern, but because it establishes a higher level of abstraction in dealing with lists. In the original definition of SCALE-LIST, the recursive structure of the program draws attention to the element-by-element processing of the list. Defining SCALE-LIST in terms of MAP suppresses that level of detail and emphasizes that scaling transforms a list of elements to a list of results. The difference between the two definitions is not that the computer is performing a different process (it isn't) but that we think about the process differently.

I declare under penalty of perjury that the foregoing is true and correct.

Dated:_____ _____
HAROLD ABELSON