We use the notation of Shanks's review [1] of Fröberg's recent table. The author has evaluated S_p for the first 21100 primes $p \equiv 1 \pmod 6$, that is for $p < 509757$. (Fröberg, at about the same time, had gone to $p < 200000$.) The number of S_p in the intervals I_1, I_2, I_3 are 5748, 6933, 8419 giving proportions 27.2%, 32.9% and 39.9%, respectively. Since $27.2\% > 4/15$, this goes to support Shanks's scepticism about Fröberg's conjecture that the asymptotic proportions should be 4 to 5 to 6.

There are two tables. The first gives the number of primes p in each consecutive 100 for which S_p is in I_1, I_2, I_3 together with the cumulative totals and their proportions (to 6 significant figures). Although there are minor fluctuations the proportion of S_p in I_1 has a rising upward trend and the proportion in I_3 has a decreasing trend. The second table lists for $n = 100000(100000)500000$ and for $x = -1(0.05)+1$ the proportion of the primes $p < n$ for which $S_p < (2p^{1/2})x$ together with the value $1/2 + (\arcsin x)/\pi$ to which it would tend under the hypothesis of equidistribution. To the naked eye there is quite a good fit but perhaps this is not a severe test.

In the introduction the author briefly discusses the method of computation and how it was organized to minimize computer time. The residue r_x of x^3 modulo p is computed by the recurrence relation

$$r_{x+1} \equiv 2r_x - r_{x-1} + 6x \pmod p, \qquad 0 \leqslant r_{x+1} < p.$$

The value of $\cos(2\pi r_x/p)$ is then computed by interpolation between the values $\cos(2\pi j/1024)$ with integral j. This requires $O(p)$ calculations for each p, but so, as the author points out, does the reviewer's method [2]. As a check she has considered the intervals investigated by the reviewer and points to a discrepancy of one unit in one of his tables. The computations were done on a BESM-6 and in all required machine time "of the order of 24 hours".

J. W. S. CASSELS

University of Cambridge
Cambridge, England

1. C.-E. FRÖBERG, "Kummer's Förmodan," *Math. Comp.*, v. 29, 1975, p. 331. UMT 5.
2. J. W. S. CASSELS, "On the determination of generalized Gauss sums," Arch. Math. (Brno), v. 5, 1969, pp. 79–84.

35 [12].— RICHARD V. ANDRÉE, JOSEPHINE P. ANDRÉE & DAVID D. ANDREE, *Computer Programming: Techniques, Analysis and Mathematics*, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1973, xvii + 549 pp., 24 cm. Price $12.95.

Despite the title, this text is largely an elementary introduction to FORTRAN programming. The first 220 pages provide a rather leisurely introduction to basic FORTRAN: integer and real variables, arrays, DO loops, and the elementary statement types. The approach is to work largely from example problems, motivating the need for each language feature and, at the same time, studying in detail the various difficulties that arise in problem formulation and analysis prior to coding. The remainder of the book consists of chapters on simulation, random number generation and Monte Carlo techniques (60 pages), errors in numerical computations (50 pages), FORTRAN subprograms (40 pages), and assembly language programming on the IBM 1130 (45 pages), followed by 100 pages of answers to problems.

The strength of the text lies in an interesting collection of problems (each chapter is based on a set of example problems and also ends with an extensive problem set). Also important is the early and continued emphasis on careful problem analysis before coding begins, a stage that beginning programmers too often bypass in their rush to get something "on the machine".

The major weakness of the text (a weakness that unfortunately overshadows the strengths mentioned above) is the extremely narrow and rather dated view of computer programming. In this respect, the text might well have been written ten years ago. Sub programs are mentioned only briefly toward the end of the book, and almost no hint of the central role of subprograms in programming is given. The sort of problem analy sis and program design suggested to precede coding is entirely concerned with questions of *run-time efficiency*—no emphasis is given to good program structure, readability, doc umentation (in spite of statements to the contrary in the text), or modifiability. The text is sprinkled with suggestions for writing programs, but most are of debatable value and in conflict with current ideas, e.g., "use the faster forms whenever feasible: ... $C*C*C*C$ is faster than $C**4$ which in turn is much faster than $C**4.0$" (p. 422), "make a simple case run first, then make it fancy" (p. 184). The beginning programmer studying this book is far too likely to get the impression that "efficiency" is not just the primary, but almost the only, criterion by which program design is to be judged.

In addition, *assembly language* is presented as the way to get more power and ver satility, should FORTRAN prove too restricted; there is only the merest mention of other high-level languages. The book has a number of other oddities: A six-page section on "The Computer in our Society" included in a chapter on "Subscripts and DO Loops", 45 pages devoted to the details of assembly language programming on the IBM 1130 (a small pre-IBM 360 machine), and the mixing of FORTRAN language details, job control language statements, particular hardware restrictions (six-digit accuracy is taken as standard), and other trivia.

In sum, an instructor might glean some interesting problems and examples from this text, but it is not recommended as a primary text for student use. Basically, the contents reflect practices, languages and computer systems of 1964 rather than 1974. A beginning student would be better served by a text incorporating more recent practice in programming.

T. W. PRATT

Computer Science Department
University of Texas at Austin
Austin, Texas 78712

36 [12].—FREDERICK W. WEINGARTEN, *Translation of Computer Languages*, Holden-Day, Inc., San Francisco, Calif., 1973, xi + 180 pp., 24 cm. Price $9.95.

The title of this book might lead one to infer that it treats compiler construction rather generally. The title is deceptive, however, in that the book is devoted almost exclusively to an expository presentation of different parsing methods and to the essential aspects of the theory underlying them. Such matters as run-time organization are not even mentioned, and code generation is given only the most cursory treatment. Thus, the book is not really a satisfactory text for a compiler course, although that would seem to be its intent.

After some preliminary introduction to relevant mathematical notation, the book discusses some of the early methods used for translating arithmetic expressions. It then goes on to a more general discussion of general formal grammars, context-free grammars, and the structure of translation trees. The author has chosen to represent such