ED 094 699 IR 000 880

AUTHOR Archibald, J. A., Jr.; Katzper, M.
TITLE On the Preparation of Computer Science Professionals in Academic Institutions.
PUB DATE May 74
NOTE 7p.; Paper presented at the National Computer Conference (Chicago, Illinois, May 6-10, 1974)

EDRS PRICE MF-$0.75 HC-$1.50 PLUS POSTAGE
DESCRIPTORS Academic Education; *Computer Science Education; *Educational Needs; *Information Science; Information Scientists; Professional Associations; Professional Education; *Undergraduate Study

ABSTRACT

A study which was made concerning the training of professionals in the computing industry found that existing programs in computer science at academic institutions do not provide the preparation required for the practice of computer science in industry. The problem is due primarily, but not exclusively, to the failure of academic institutions to consider the needs of industry in ordering their priorities and goals. Other causes are the specific failure of industry to make its needs known to academic institutions and the general lack of communication between these communities. Several progressional societies are trying to bridge the gap with programs for the certification of professionals. Re-ordering of priorities within undergraduate programs and an application-oriented undergraduate program are presented as a means of solving part of the problem. Specific suggestions of other steps that may be taken in the industrial and academic communities to strengthen the profession at large are also discussed. (WCM)

On the preparation of computer science professionals in academic institutions

by J. A. ARCHIBALD, JR. and M. KATZPER*

State University of New York
Plattsburgh, New York

INTRODUCTION

One of the major problems facing the computing industry concerns the training of professionals. Many of the existing programs in computer science at academic institutions simply do not provide the preparation required for the practice of computer science in industry.[1,2] This problem is due primarily, but not exclusively, to the failure of our academic institutions to consider the needs of industry in ordering their priorities and goals. Other causes are the specific failure of industry to make its needs known in an adequate manner to the academic institutions, and the general lack of communication between these communities. Regretfully, neither community is taking adequate steps to solve the problem. In this void of activity, we find the several professional societies trying to bridge the gap with programs for the certification of professionals.[3]

This paper discusses the nature and implications of the situation, calls for a reordering of priorities within undergraduate programs at academic institutions, presents an applications oriented undergraduate program in computer science as a means of solving some parts of the problem, and makes specific suggestions of a number of other steps that may be taken in the industrial and academic communities to strengthen the profession at large.

THE PRESENT SITUATION

The disparity in the nature of computer science, as practiced in industrial and academic institutions, is abundantly clear to anyone who has participated in both areas. A major portion of the difference is the age-old dichotomy between the abstract and the applied. The primary requirement in industry is for applied computer scientists—people who can interact effectively with practitioners of other professions to develop meaningful solutions to existing problems within the limitations of present day facilities. The primary products of an academic institution are the theoreticians, capable of conceptualizing solutions to formally posed problems and relating these solutions to computational processes. Said in other words, industry needs people with applied backgrounds and academia is producing people with theoretical backgrounds.[2] Thus, the priorities are different. The industrial priority cannot be criticized, since it has passed the one universal industrial requirement—profitability. This profitability is the cornerstone of a thriving, new profession. The academic priority is also not wrong—it is the essential ingredient to the future of the discipline. The point is, however, that academia needs to recognize not only the priorities for the future, but also those for the present. It must increase its emphasis upon applications without sacrificing its emphasis upon theory. This can be done by tailoring the undergraduate program toward applications, and retaining the theoretical emphasis in the graduate programs. Undergraduate curricula should be geared to the majority that do not proceed on to graduate school rather than to the minority that do.

The fact that this dichotomy between theory and applications exists is not, of itself, bad. Indeed, there are many disciplines in which the divergence is greater than it is in computer science. The problem is one of timing. Our discipline was formed out of necessity. It is thriving because it was born with a "golden spoon in its mouth"—a large backlog of present day problems capable of solution within existing theory on existing facilities and literally crying out for solution now, in the present. The solution of these problems should enjoy a higher priority in our profession (particularly on academic campuses) than it presently has. Said in other words, our discipline is too young, and the pressures upon it to solve existing problems are too great for us to be able to afford the luxury of the current divergences between theory and applications. Divergence within the practice of a discipline is healthy and desirable once the discipline itself has matured. Ours has not, and will not mature until a sizable dent has been made in the backlog of problems.

The industrial world has only recently awakened to the potential of a computerized age—an age in which machines can and are both performing routine tasks of drudgery, and

* Presently with Ocean Data Systems, Inc., Rockville, Maryland.

providing formerly incomprehensible solutions to complex, practical problems. An age in which, however, computers are doing less than they should. The great challenge of industry is to harness the power of the computer in the service of mankind. Within industry, there is a need for people (primarily at the Bachelor's level) with the capability of joining and contributing to the problem solving team through the infusion of new ideas, techniques, and capabilities. This means people who have some understanding of both the problems of the industry concerned, and the methods of using currently available computers to solve these problems. This, in turn, means people who have combined a strong background in the area concerned with an ability to actually analyze problems, design and implement programs, and perform the other tasks related to the effective use of computers. These individuals must be versed in the various areas of computer techniques and analysis: e.g., statistics, simulation modeling, numerical methods, non-numerical methods, artificial intelligence and the like. They must understand such things as the effective organization of data, the effective utilization of processor time and memory, and the nature of multiprogramming computer systems. They must have a degree of both competence and sophistication in programming—the primary skill of computer science. These must be gained through in-depth training (in both classroom and laboratory situations) at the level of the current state of the art. In short, they must be competent applied computer scientists.

Our academic institutions have also, recently, awakened to a new age. They, too, have become impressed by the great potential of the computer. Their reaction to the challenge has been to plunge headlong into the development of advanced theories and courses which teach these theories, without concern for the practical uses of either the existing or the developing theories. Indeed, by virtue of their background, many have been too concerned with the new theories, and too disinterested in the practical uses of today's theories. The ACM Curriculum 68,[4] upon which many of our academic programs are based, contains numerous courses which use the current state of the art merely as a point of departure to enter upon abstract studies. These courses contain little emphasis or practice in the use of present technology, with the limitations of our present facilities. Thus, many of the young people who leave our academic institutions possess an outlook and orientation which are unsuited for doing the practical work expected of them in industry.[6] Industry has reacted to this situation by hiring college graduates in their major area of interest. These people are then hastily trained to do some programming. The results are, naturally, less than professional.

Another shortcoming of present academic programs is the tendency to concentrate computer science programs exclusively upon the computer, as though the computer were an end in itself. Hence, our graduates have little idea as to what the theories they have studied are good for. R. W. Hamming has referred to these non-professionals as "idiot savants" and "computniks."[5] Hiring such a graduate in computer science has become a burden that few of our industries with problems to be solved can afford.

REASSESSMENT OF PRIORITIES

The intent of this paper is to encourage the de-emphasis of theoretical computer science at the undergraduate level. Theory is important, for without it, the discipline will die, and many of the problems of the future will go unsolved. The plea is for a reassessment of priorities. Much more emphasis needs to be placed on applications in the undergraduate curriculum, with the emphasis remaining theoretical in the graduate curriculum. This re-directed emphasis on applications in the undergraduate program should have two thrusts: (1) the development of methodologies to be used in solving general classes of applied problems, and (2) the development of a level of competence in one or more of the disciplines to which computer science is applied.[6] We do, indeed, note some movement in this direction.[7] Our position is that this movement is insufficient. Under the present system, many of those who complete the undergraduate program are really unqualified for anything except graduate school. They go on to complete their graduate degrees with an esoteric thesis and little exposure to applications. Their training leads them to seek positions as undergraduate college instructors—they are unprepared for industry. As instructors on the undergraduate campus, they participate in the training of the next generation of undergraduates—again without exposure to applied computer science. And so, the vicious circle is closed. It continues to roll on by the sheer force of its own momentum.

This is not to take away from the importance of developing new theories. Indeed, such development is the vitality of a thriving discipline. Retention of theoretical emphasis in the graduate curricula should meet these needs. In other disciplines, where new problems are being solved in a timely manner as they occur, and where there is no backlog, an academic establishment concerned solely with speculative theories is ideal. However, in computer science, where there is a backlog, it seems appropriate to ask that the academic world devote a significant part of its effort to the present, the practical, and the applied. In short, there is a great need for the academic institutions to divert a greater part of their efforts to applied computer science. Theory can find its place in such a program in support of applications, in deepening understanding and in broadening horizons for imaginative solutions to problems. At the present, however, much theory is presented for its own sake. Significant applied and theoretical problems are being discarded for vacuous speculation. The result is that our academic institutions are not producing the kind of talent required by our industrial institutions.

The question as to whether academic institutions should produce people to meet specific demands of society, or merely educate people to enrich their souls, is an age-old question which will merely be mentioned here. We do live in a world of practicality, a world in which one must eat, a world in which the individual who can serve society has a higher survival potential than the individual who cannot.

THE VIEW FROM INDUSTRY

Industry, of course, is not to be denied. For one thing its position is vindicated in our society simply because its survival is based solely upon its ability to make profits. Industry will pay to get what it wants, or what it thinks it needs, in terms of manpower. It trains its own people when the skills it wants are not available. The problem here is that industrial training, by being oriented around corporate profits is generally inadequate, and usually falls far below professional standards.

Industry certainly needs people with, (1) an understanding of industrial problems, (2) an understanding of computers, and (3) fresh new ideas. Regretfully, the academic institutions are not producing sufficient numbers with this combination. As a result, industry hires graduates with majors in their areas of interest, and hastily trains them in computer skills. These graduates arrive at industry with, at most, two of the three major qualifications. The third one is developed, on the job, in a hit or miss manner. Industrial training does, indeed, tend to be very shallow. It is designed to meet certain very narrow objectives. Persons trained in industry are usually trained for a very specific task—with neither breadth nor depth. Such people must be continually retrained whenever their skills are required in a different activity. Industrial training is not geared to the production of professionals. These new employees certainly learn to program, but they tend to be weak in the areas of non-numeric methods, statistics, and the like. When an unusual problem arises they are not able to cope with it. This programming staff is often supplemented by graduates from two-year schools, who are often programmers with little understanding of more than straightforward coding, and graduates from the various training institutes. The training institutes themselves, however, are for the most part, designed to train technicians. The result is a total industrial staff that is less effective than it should be. It is distinctly subprofessional.

AN ACADEMIC RESPONSE

There is much that an academic institution can do to meet the needs of society. The new Bachelor's program in Computer Science at Plattsburgh State University College, included as an addendum to this paper, is one approach to this problem. Under this program, our computer science majors are required to develop a solid ability to do effective programming in at least two languages (one of which must be FORTRAN), to study the various areas of application and methodology of problem solving such as systems analysis, simulation, statistical methods, numerical methods, and non-numerical methods, obtain a modest theoretical background, and take an area of concentration in a discipline which uses the computer.[5]

The intent of this paper is not to present the program at Plattsburgh, but rather to encourage other institutions to devise their own approaches to applied computer science at the undergraduate level.

Specifics are indeed important at this point. Some of the features of the Plattsburgh program which, in the opinion of the designers are both important and unusual are listed below. We do consider these items to be essential in the proper training of computer science professionals.

Training in programming skills

We note an unfortunate disdain for courses whose primary objective is the teaching of programming skills in specific higher level languages. As stated above, programming is the fundamental skill of the computer scientist. We do not anticipate that our graduates will spend the major part of their time in actually writing instructions for a machine. We do feel, however, that skill and ability in the area of programming are essential if an individual is to perform any task related to the use of machines to solve problems effectively.

The first course in the major sequence at Plattsburgh concentrates primarily on the development of programming skills in FORTRAN. We use this language, despite all of its actual and theoretical deficiencies, because it is the primary language of the industry, and because there seems to be little chance that this situation is about to change in the foreseeable future. Our students are given instruction in the language itself as well as general instruction in computer science and programming. They receive numerous programming assignments, and, in addition, are required to complete an independent project in some area of interest to themselves. This course has a reputation of being difficult. There are numerous suggestions that additional academic credit be assigned to this course. Our experience with this course makes us wonder how effective programming and the numerous other topics that many institutions offer in their first course can be adequately covered in a single course. We use a tough two-semester sequence to offer, in greater depth, and with more actual programming on the computer, that which many other institutions offer in one semester.

We also have a similar course in COBOL programming. Most of our students use COBOL as their second language. Students in this course are given general information in computer science and programming, and specific instruction in COBOL, with emphasis upon the latter. Numerous programming assignments are also given in this course.

Area of concentration outside of computer science

The fact remains that computing is not an end in itself. Man does not compute for the sheer sake of computation, but rather because he is interested in something else—something in which computation is used to further understanding. Our students are expected to select some outside discipline to which computers may be applied, and to gain a depth of understanding that, while less than that of a major in the discipline, will be sufficient to permit the graduate of our program to interface with professionals of the other disciplines in the solution of problems of that discipline. Included in the possible areas of concentration are the natural, physical, biological, mathematical, social, behavioral, linguistic and administrative sciences. We have also included disciplines where the relationship is not quite as obvious, such as art and music. This is the feature of our program that, we feel, is a direct reply to Dr. Hamming's comments.[5] We note that certain other schools attempt to solve this problem by offering only graduate degrees in computer science.

Courses in techniques and methods

Our program includes courses in specific techniques and methods, such as statistical methods, numerical methods, non-numerical methods, systems analysis, and simulation and modeling, in which the emphasis is placed upon actual problem solving. Programming per se in FORTRAN is not stressed in many of these courses. We assume a background from previous courses sufficient for carrying out programming assignments. We stress the development of specific techniques, the use of specialized languages designed for the area concerned, and the effective use of library subroutines to solve specific types of problems of interest. Thus, the focus is upon the effective utilization of the computer, through whatever means are available, to solve problems of concern.

Courses in the fundamental nature and theory of computers

We do offer courses in machine and assembly language programming, programming languages, operating systems, and discrete structures. Certainly, no major sequence would be complete without courses in these areas. Our courses emphasize applications as much as possible. For example, in programming languages, we study the languages themselves, and their underlying structures from the perspective of how the language is used to solve the type of problem for which it was designed. Thus, for example, in the study of SNOBOL, we try first to understand the nature, applications, and problems of string processing, and then study the structures of the language as they relate to string processing. We do spend some time in a general discussion of abstract language structure, but we do not emphasize such considerations.

RECOMMENDATIONS FOR INDUSTRY

So far, we have focused our presentation on our sister academic institutions. Let us now consider the role of industry in meeting the problems of training the computer professional. If the academic institutions have been too unapplied, and too theoretical, certainly the reverse is true of industry. Too often the sights of industry have been narrowed to the area where the industry presently thinks that it can be profitable. These decisions of likely profitability are generally short range.

In general, industry is better able to carry out large scale advanced application developments. Industry has been deficient in both doing the application development work and in inspiring the academic institutions to do this work. Industry should interact more strongly with the academic world and, in doing so, get the academic institutions back on the track of reality. Specific suggestions (some of which have been discussed in the past[1]) include the following:

A. Providing of more grants to academic institutions to solve specific existing problems. Frequently, industry has the funds necessary, but neither the actual nor potential talent to solve its major problems. Academic institutions have few funds, but possess great potential for developing the necessary talent to solve these problems. The funds contained in a grant from industry would motivate the academic institutions to develop the needed talent. Academic institutions badly need the encouragement of industry to turn toward applications, and the availability of grants for specific purposes is the best way of providing this encouragement.

B. Providing computer time to academic institutions for use in developing techniques and methods for the solution of specific existing problems. Actually, this is another form of the grant suggestion made above. Relatively few academic institutions have the facilities needed to handle significant applied problems. This is particularly true in the area of scientific applications. Industry, by making its excess time available, and by indicating the kinds of problems that they are interested in having solved on the machine, can very effectively re-direct the academic institutions to significant applications.

C. Temporary employment of academic faculty members. One of the major causes of the current problem is the lack of communication between industrial problem solvers and academic faculty. There needs to be more contact between these groups. There should be a flow of both ideas and people between the two communities. Each community needs to know the problems of the other. Faculty members have always been available for temporary positions for durations of several months (one summer) up to fifteen months (two summers and the included academic year). The nature of college teaching lends itself to special summer work, and to sabbatical leaves. Very often, temporary employment for these periods is obtained at other academic institutions. Industrial institutions should avail themselves of the services of faculty members as consultants and in other special assignments of a temporary nature to aid this flow of ideas.

D. Making industrial personnel available to academic institutions either as temporary full-time faculty members or as part-time adjunct faculty members. This suggestion complements the previous one, and it has the same motivation, the flow of ideas and people between the academic and industrial communities. The temporary use of people on a full-time basis is the reverse of the academic sabbatical. It would be of great help in making the academic institutions cognizant of the problems of business, while enabling the academic institution to retain the flexibility of staffing that is so important in the modern college.

E. Providing "internships" with pay for students. This again may take one of two forms, the full-time employment of students in areas of interest for specified periods of time, and the part-time employment of students in areas of interest for longer periods. Certainly the best way to learn applications is to be involved in applications. This experience can be very meaningful in directing students to promising areas of application, and would be an ideal supplement to the instruction in methodology and techniques. Additionally, it would provide the student with badly needed funds. The actual employment, if over the summer, would be analogous to the conventional summer job. Full-time employment for a single semester would be analogous to the semester of student teaching that education students go through, or the internship served by new medical doctors. It would have the same benefits. Where conditions permit, half-time employment, and half-time course loads is another very viable possibility.

CONCLUSIONS

We, collectively, as practitioners of computer science, at both industrial and academic institutions, have a serious problem in terms of effective utilization of human resources. We must all dedicate ourselves toward working together to solve it if we are to make the contributions to civilization that we have been challenged with.

ADDENDUM—The Applications Oriented Undergraduate Program in Computer Science at Plattsburgh State University College

General Implementation

Students in the computer science major program at Plattsburgh are required to take a broad spectrum of applied computer science courses which include techniques and methodologies, a kernel of pure computer science courses which include concepts and theories, and a sequence of courses in an area of concentration, or a discipline with present or potential computer application. In our consultations with our students we point out the interactions between computer science and other disciplines. As a result of our applications oriented philosophy, our students are encouraged to devise study plans that combine investigations in computer and information science with the study of other fields. Simultaneously, they learn to apply the ideas of systems analysis to areas of contemporary concern. We emphasize the sequence in a related discipline as being the specific means of avoiding the lack of breadth which was the basis of much of the historical opposition to undergraduate programs in computer science.[1-5] Many of the courses included in our program are designed for both computer science majors and for students needing courses to support work in other disciplines.

The faculty at Plattsburgh is unusual in the large degree of interdisciplinary collaboration that goes on in research and teaching. This factor has aided greatly in the implementation of our computer science program. Our colleagues have joined us in advising students as to how to apply the knowledge obtained in computer science to yield interesting results in other disciplines. Concurrently, we have joined our colleagues in aiding them to include the use of computers in their own courses, and in their research. Many of the applications advocated at computer science education conferences and in the literature have been independently implemented at Plattsburgh.[3,9,10]

Specifics of Implementation

The emphasis in our computer science curriculum is upon applied computer science, i.e., upon the use of computer science to support work in other disciplines. Within the scope of applied computer science are both numerical and non-numerical applications. Numerical applications include numerical methods, statistical analyses of data, simulation and modeling, and applications to business and the quantitative sciences. The non-numerical applications include list processing, string manipulation, text editing, information storage and retrieval, and applications to the graphical sciences. Applied computer science is also involved in the mastery of techniques for organizing complex informational structures and the development of skills in analytic approaches to problems. Such skills are especially developed in the study of systems analysis and artificial intelligence.

There are a number of courses which we denote as pure or theoretical computer science courses. In order for any one discipline to be used effectively in support of other disciplines, its own development must have a certain degree of sophistication and maturity. Thus, the practitioners of computer science must have a certain competence in pure or unapplied computer science in order to apply it to advance other disciplines. Pure computer science includes such things as programming in machine and higher level languages, the study of languages and compilers, operating systems, discrete structures, computer logic, computer electronics and theories of computability and automata. We expect our majors to acquire a degree of competence in these areas.

In Table I, we present a general overview of the courses in our program. We note that the exact sequence for each student is determined on an individual basis depending upon the interests and capabilities of the student concerned as well as the area of concentration selected by the student.

Descriptive Listing of Generally Recommended Courses

Introduction to Electronic Data Processing: Introduction to computers, information science, data processing, concepts of hardware and software, cybernetics, and programming in BASIC language. Major emphasis on concepts of information science rather than on computer programming.

TABLE I

Usual Year of Study in the Computer Science Program	Courses
Freshman	Introduction to Electronic Data Processing (with BASIC) Systems and Information
Freshman or Sophomore	Introduction to Computer Science I & II (with FORTRAN) Second Higher Level Language Computers and Society
Sophomore or Junior	Machine and Assembly Language Programming Discrete Structures and Computational Analysis Numeric Methods Non-Numeric Methods Analysis of Statistical Data
Junior or Senior	Programming Languages Operating Systems Graphical Methods Artificial Intelligence Simulation and Modeling Undergraduate Independent Study Undergraduate Research

Systems and Information: The course will show how to study complex systems in terms of their interacting components. A general approach to such studies will be presented. This approach will involve analysis, modeling, and simulation of systems together with information flow, control, and information processing within systems. Interdisciplinary skills are used for analysis and modeling in the social sciences, natural sciences and environmental studies. Contemporary examples will be taken from areas of concern such as ecology and economics. The role of information flow and control in systems is investigated. The processing of information by computer is demonstrated. An individual term project requiring the student to analyze a system of his choice is included.

Introduction to Computer Science I (with FORTRAN): Introductory course in computer science. Introduction to computers and use of computers to solve problems. Emphasis on introduction to FORTRAN programming. Numerous programming assignments. Development and implementation of independent project of the student's choice is required.

Introduction to Computer Science II: Intermediate problem analysis and FORTRAN programming, introduction to error analysis, numerical applications, non-numerical applications, operating systems, and machine and assembly language programming, survey of higher level languages and debugging techniques.

Second Higher Level Language: Introduction to computers and the use of computers to solve problems in the language concerned. Emphasis on programming in language concerned.

Machine and Assembly Language Programming: Elements and techniques of machine and assembly language programming as applied to hypothetical and actual computers, with emphasis on relationship to programming in higher level languages.

Non-numeric Information Processing: Fundamentals of non-numeric information processing, and programming techniques used in the solution of non-numeric problems. List and string processing languages and compiling techniques. Emphasis on applications.

Discrete Structures and Computational Analysis: An introduction to discrete mathematical structures with special emphasis on theories that are relevant to the study of computer science. Topics include aspects of logic, Boolean algebra, finite machines, switching circuits, set theory, induction, trees, strings, graphs and finite automata. Emphasis on applications.

Numerical Methods: An introduction to the basic notions of numerical solution of problems through an in-depth study of numerical processes. Emphasis is placed on errors inherent in computation and error analysis.

Computer Analysis of Statistical Data: An introduction to the use of a digital computer for the analysis of statistical data. The characteristics and significance of common statistical tests and distributions are studied through experimentation on a computer. Existing computer-library routines are used where possible. Application to problems in the student's major area is stressed.

Simulation and Modeling: Introduction to the use of computers for the simulation of systems studied in the fields of natural, social, and administrative sciences. Emphasis is on the formulation of problems and the design of models which reflect the activities of the systems concerned, and which lend themselves to being programmed for the computer. Probability functions, stochastic variables, Monte Carlo techniques, and queuing theory. Both continuous and discrete systems are studied in detail. Specialized simulation languages and statistical verification of simulation results are also considered.

Programming Languages: A study of the general principles and applications of computer languages and the techniques and problems of language implementation. Emphasis will be placed upon the use of significant languages excluded in other computer science courses, e.g., ALGOL, PL/I, APL, LISP, and SNOBOL.

Operating Systems and Systems Programming: Detailed study of the design and nature of the interface between man and computer: job scheduling, resource allocation, task overlapping, input/output handling, interrupt processing, multiprogramming, multiprocessing, time-sharing, assembly and compilation, and system subroutines.

Graphical Data Processing: Nature of graphical information processing and programming techniques used in the solution of these problems.

Artificial Intelligence: A survey of the application of heuristic programming techniques in computer game playing, problem solving, searching and pattern recognition. Principles involved in list processing languages will be reviewed. Machine learning approaches will be used for the writing of interactive educational programs.

Independent Study and Research: Project or study individually arranged between student and sponsoring faculty member. Results presented in seminars.

SUMMARY

We feel that the sum of the courses and requirements presented above yields more than the knowledge of a computer as a problem solving tool. Rather it imparts an ability to recognize and formulate problems, to seek solutions to these problems, and to make optimal use of the computer. This results in part from the emphasis that the faculty lends to the subject matter, and the interaction between students and faculty. A graduate with a bachelor's degree (in any field) will hold a subordinate position in his first job. As such, he will be carrying out tasks assigned by others. As a result of our program, our graduates will have an understanding of the requirements and interests of their superiors and will be able to carry out their duties in an appropriate and efficient manner.

REFERENCES

1. "Higher Education: Study Urges Altered Thrust in Federal Support," *Science*, Volume 182, Number 4112, November 1973.
2. "The Misdirection of Computer Education Efforts," *Journal of Data Education*, 1972.
3. Bylaws, Institute for Certification of Computer Professionals (ICCP).
4. Curriculum '68," *Communications of the ACM*, Volume 11, Number 3, March 1968.
5. Hamming, R. W., "One Man's View of Computer Science," *Journal of the ACM*, Volume 16, Number 1, January 1969.
6. Kandel, A., "*Computer Science—A Vicious Circle*," *Communications of the ACM*, Volume 15, Number 6, June 1972.
7. Austing, R. H. and G. L. Engel, "*A* Computer Science Course Program for Small Colleges," *Communications of the ACM*, Volume 16, Number 3, March 1973.
8. Blum, R., Ed., "Computers in Undergraduate Science Education," *Comm. on College Physics*, 1971.
9. Proceedings, *Conference on Computers in the Undergraduate Curriculum*, 1970-1-2-3.
10. Recommendations for an Undergraduate Program in Computational Mathematics, Committee on the Undergraduate Program in Computational Mathematics, May 1971.