OXFORD UNIVERSITY COMPUTING LABORATORY
PROGRAMMING RESEARCH GROUP

Professor of Computation: C. A. R. Hoare

45 BANBURY ROAD
OXFORD OX2 6PE

Tel. Oxford 58086

Prof. M. Rogers
Bristol.

22 June 1982.

Dear Mike,

You asked me to put in writing my comments on your draft proposal for BSc honours in Computer Science.

I very much like the first sentence of the philosophy. But I would continue:

"The computer scientist uses the technique of levels of abstraction to master the complexity of this subject and its applications.[2] The subject itself is studied at many levels of abstraction, and the computer scientist must understand them, and the relations between them. At the same time, it is a practical subject, and at all levels the computer scientist must be able to translate

his understanding into skills in inventing, design, verification, implementation of useful computer systems.

"The basic levels at which the subject are studied are:

(1) Digital logic.
(2) Computer Architecture
(3) Programming in a low level language
(4) Programming in a high level language
(5) Program Specification
(6) Theory of Computation.

" "

The bridging subjects between these levels are.

(1)-(2) VLSI chip design
(2)-(3) Assembler & basic software
(3)-(4) Compiler Construction
(4)-(5) Program design and verification."
(5)-(6) ? Formal language Definition.

At the same time, certain important application areas must be studied:

(1) Scientific / engineering calculations including Numerical Analysis.

(2) Commercial Data Processing including systems analysis

(3) Operating Systems.

(4) Communications.

On CS1A. I would emphasise the <u>skills</u> which I wish to impart:

e.g. ① to design and write PASCAL programs up to 120(?) lines.

② to read, understand, and modify well-documented PASCAL programs up to 600 (?) lines.

③ to understand and use effectively the available simple programming tools.

Similarly, for unit 3, where it seems that the storage management could well be omitted.

Similarly for the course in Discrete mathematics, which should have as its objective that the student should be able to <u>prove</u> theorems using algebraic identities and natural or structural induction; and that he should be able rapidly to find <u>counterexamples</u> for "proposed" theorems which are false. For this reason I would include predicate logic in the syllabus. The "size" of the ~~the~~ proof should be limited to about ten lines; but he should be able to read and check proofs up to (say) thirty lines.

I would combine courses (b), (c), (d) into a single course with a practical orientation towards VLSI design.

In general, it is administratively very convenient to introduce a standard sized module for your course (say 36 lectures). Then each course can have a single examination set ~~by~~ and marked by a single examiner. You will also find that your syllabus is much more flexible, and you can rapidly develop and improve the syllabus. Within each standard module, the lecturer ~~has~~ has considerable freedom of improvement; and it is much easier to drop old courses and to try out new ones.

Furthermore, particularly for the more advanced and application-oriented courses, you will probably want to offer some choice. It's surprising how tastes differ, & e.g. between those who are excited by the theory, and those who take to commercial data processing.

Thanks again for your hospitality and good luck in your course development.

Yours sincerely

Tony