A computer program for banking multiple choice questions

M. D. Buckley-Sharp* and F. T. C. Harris†

* *Bland Sutton Institute*
† *Department of Biology as Applied to Medicine, 'Medical Education Group', The Middlesex Hospital Medical School, London W1*

A new technology of teaching and learning in higher education is developing. This technology, based in large part on multiple choice examining, is already producing important information on the mechanisms of vocational University Education.

At the Middlesex Hospital Medical School, the Research and Services Group for Medical Education is already collaborating with over 70 Departments in Faculties of Medicine throughout Great Britain. A very important part of the information generated by the procedures operated by the group relates to the multiple choice questions themselves and their associated test statistics. In order to manipulate and retrieve this data the Group have developed programs and sets of procedures. This question Bank system is complementary to existing procedures for scoring and analysing examinations. The question Bank program accepts data on cards, or magnetic tape, converting it to a suitable format for a random access file on magnetic disk. A command language structure enables the information to be manipulated with the minimum of coded instructions.

The system is capable of expansion, and also of diversification to other fields, but the optimal level, and type, of computer hardware for its operation is not at present available for use.

(Received December 1969)

There is a growing awareness of the need to sharpen the tools of assessment in higher education. To an extent, this need to develop new techniques of educational 'quality control' is forced on us by the complexity and cost of higher education. This process needs to be applied both to course, and to student assessment, and the combination of the two has led to greatly increased demand on the available facilities. For several reasons, multiple choice questions (M.C.Q.) have been used. Most people are familiar with M.C.Q. as a type of question in which a question-stem is followed by a number of possible answers, the examinee being invited to choose, from among the answers, those which fit the stem. Various types and formats of question have been devised, all with similar aims, and similar properties. M.C.Q. are used in the examination (assessment) process, but they also have wide appeal as teaching aids, and in opinion surveys.

Several computer programs have been devised to score the data arising from the application of M.C.Q. (e.g. Groves, 1968, Anderson, Wood, and Tomlinson, 1968, Harris and Buckley-Sharp, 1968), and particular interest has been shown, and much work carried out using these techniques, in the field of medical education. A major part of the procedure, not covered by these systems, is the storage and retrieval of the questions themselves, together with the statistical data which arises from their use. A system which uses a computer program (QBANK3B) as its focal point has therefore been devised and set up over the past two years to deal with the problems associated with such storage and retrieval. Some idea of the problems involved can be gained from the fact that our current question bank comprises nearly six-thousand questions, each held on an average of ten cards. This bank is growing at nearly four hundred questions per month, making the checking and updating of the data a time-consuming process.

The details of the parts of the procedure external to the computer program have been discussed separately (Buckley-Sharp and Harris, 1969), and they will not be discussed here. It is the purpose of this paper to outline the program itself, and its method of operation. Nominally, the program will manipulate up to four disk volumes, one at a time, giving access to nearly 30×10^6 characters of data per single program application. Different program applications may use different volumes giving unlimited storage access.

The problems of the system design

In a data banking system, the definition of the file structure and the precise format of data records are the prime considerations. It does not follow that the format of the initial data, e.g. on cards, should be retained as the format of records on accessible, or back-up, media: this may be a most uneconomical proposition. However, variations in format introduce programming complexities.

The unit of information is the record, and only occasionally will it be necessary to obtain discrete data from within records, for listing. This of course is in contradistinction to the program requirements for data, in which sort-keys, block counters, flags, and pointers of all descriptions need to be maintained to monitor the state of the system. In the interest of simplicity, the access procedures available to the user should not need to consider this multiplicity of control information, and a mnemonic language, or command structure is ideal. Command structures are familiar in very high level languages, and in the procedures used by operators to control the main computer system.

Question format: mode of access

Questions are stored on 80-column cards as

(i) from one to forty 'text' cards
(ii) a single 'status' card
(iii) from one to forty 'data' cards. Questions without data cards will be accepted by the program, but a special dummy data card is constructed and stored internally with the question. This dummy data card is deleted when genuine data becomes available.

Each 'data' card is a product of a single run of the question in a multiple choice question test. The card is produced by the SCORE7L marking program (Harris and Buckley-Sharp, 1968, User Bulletin 11), when the required option is set. SCORE7L also produces a 'status' card for each question which is used only to check the answer pattern. Question references (Buckley-Sharp and Harris, 1969) are added manually to these cards.

Questions may be placed on magnetic tape, when the format is precisely the same as that for cards, i.e. the data exists as card images on tape, and so card, and tape, may be considered as identical media for the purpose of the Question Bank Program.

The manipulative operations of such a filing system are best conducted using random access to the large amounts of data involved. This means that the data must be available on magnetic disk rather than on cards, or tape. Translation between these various media is the function of the FILE, and DUMP commands (see Bank Control Language below).

When resident on disk, the question data is in a compressed form which enables considerable economies in access time to be made. Questions are stored on disk as

(i) a single 'status' record,
(ii) from one to forty text records,
(iii) from one to forty data records.

Note that the position of the status record is different to the position of a status card. The content is also different, in that the disk 'status' record contains various block counters to facilitate access to the remainder of the question. The text itself is in a compressed form with all strings of blanks removed. Such removal is signalled in the record by a numeric zero (two bytes), followed by a count of the number of blanks removed (two bytes), followed by the continuation of the text following the omitted blanks. All transactions to or from disk storage take place in compressed form, with the text being 'compressed' before writing out, and 'released' after reading in. The process is in fact slightly more subtle, for a preliminary survey determines whether one or more complete text records are saved by such compression. Where this is not the case, the compression-release cycles are bypassed. Considerable advantage is gained even by this simple method; the amount of disk space commonly saved being between 20% and 30%.

By the nature of the data, it is necessary to access records which correspond in size to card images (in fact, the disk records are 66 bytes long). If a file were composed of direct access records of this size, it can be estimated that half the available space would be occupied by inter-record gaps! Since the supplied software does not support blocked direct access records, it has been necessary to provide just such a facility within the program. This involves the provision of what amounts to two buffers, which are used to deliver and receive data to and from the program. The buffers correspond to twelve records (as viewed by the program), and are written and read *in toto* to or from disk to keep the disk file updated. Four of these long records are contained on each disk track, giving a total of forty-eight program records per track, or exactly double the number which could be stored in the unblocked state. Using this form of blocking, and the text compression already outlined, it has been estimated that some 120,000 program records may be contained on a single disk volume (I.B.M. 1316 for I.B.M. 2311 drive). This compares to the 44,000 which would be possible if the original cards were to be placed by themselves into a direct access file. This system also conveys the further advantage that I/0 activity is reduced. Once a buffer has been retrieved, individual 'records' may be delivered by program 'direct access' statements without further file manipulation.

In order to access the files of information when they are held on disk, a dictionary is used. Each dictionary entry has two parts; a question reference, and a record number. When a dictionary entry is being checked, it occupies ten bytes of store, but when it is not being used, some bits are redundant, and each entry is held, without loss of data, in only six bytes. This increases the number of entries which can be held in store at any one time to 12,000. The question reference (bytes 1–6 of expanded version) comprises three letters, and three numbers; a format which permits mnemonic subject classification (letters) and serial numbering (Buckley-Sharp and Harris, 1969). The record number (bytes 7–10 of expanded version) is the location within the file (starting at 1, and assuming unblocked records) of the first record of the question. Since the first record is the status record, and since the status record contains the counts of both text and data records, questions are accessed by the following procedure:—

(i) find question reference in dictionary
(ii) get associated record number
(iii) read this record and get counts of text and data records
(iv) read text records using counter
(v) read data records using counter
(vi) 'release' the compressed text ready for use.

Since a bank file is only accessable via a dictionary, the dictionary must be stored at the end of the job, or when a new file is to be used. This is done by dumping the dictionary onto the same disk volume as the bank file, and dismounting the disk. Remounting the disk, and reading in the dictionary is sufficient to restore all file access facilities. This is economic in computer time, as the dictionary occupies only some 2%–3% of the disk space need for a bank file, and therefore takes little time to recover.

Table 1

BANK Control Language: Command Function Summary

Command	Function
OPEN	Initiate a random access disk file for use.
CLOS	Terminate use of a disk file.
FILE	Enter data to a disk file, OR, bring a dictionary into core.
DUMP	Dump data from a disk file, OR, dump a dictionary from core.
EDIT	Edit the contents of a disk file.
DICT	Write out the current dictionary.
WRIT	Write out the current disk file.
WORK	Make selections from the current file using statistical profile.
KWIC	Make selections from the current file using textual profile.
STOP	Close down operations.

BANK control language

The BANK Program operates under the action of a series of commands. These are executed sequentially on recognition, and the commands themselves are not stored prior to execution. It is therefore not possible to alter the logical flow during program execution where a batch processing system is used. Nor is it possible to check syntactical accuracy of the data stream before commencing execution. If and when commands can be given via an on-line terminal, these problems will be much alleviated. Indeed, the structure of the system has been defined with just this possibility in mind. In the opinion of the authors, such data banking is best carried out in conversational mode, as the requirements for input commands, and output (console or video terminal) are small, and constitute a minor part of the operations. The devising and writing of the required software is however complex, and comparable, in terms of effort required, to operation software.

There are currently ten commands, each consisting of only four letters. The letters have been chosen to indicate the command function. They are summarised in **Table 1.**

Operands are used to supplement some of the commands. They are required when setting up a disk file, and when reading data to or from disk, and another data storage medium (cards or tape). The commands which require operands are therefore the OPEN, FILE, and DUMP commands. The operands themselves specify the location of the required data on its data storage medium. These are logical references, and do not themselves specify physical data locations. The connection between the logical and physical references is made using job control language (J.C.L.) facilities, available under the System 360 Operating System (O.S. 360) (IBM Form C28-6539). The interpretation of the first operand i.e. the restrictions placed by the program on the connections which may be established by J.C.L. are shown in **Table 2.**

Second operands are only used when the first operand has the value 02, 04, or 16. In this case, the second operand specifies the sequence number (first, second, etc.) of the data-set on the logical unit specified by the first operand. Thus a first operand of 02, and a second operand of 001 specifies the first data set on unit two. The correspondence between this specification, and a physical data location is maintained by the job control language DDNAME (IBM Form C28-6539).

Each of the ten commands, with its operands if needed, is recognised by a supervisor program which then takes the appropriate action by transfer to one or more subprograms designed to service the request. Some of these service subprograms read data cards themselves, so that commands will be interspersed with other data. Within the service subprograms, reversion to the supervisor may be automatic, or depend on the flow of data. In general, a routine which itself reads cards, will revert to the supervisor program when a card is read bearing the name of the command which caused entry to the routine. Thus, when transfer has been made to the routine which services the EDIT command, the reading of another EDIT card causes reversion to the supervisor. The commands which may read cards, and to which this applies are the FILE, EDIT, WORK, and KWIC commands. Other commands do not read data cards, and the card following one of these commands is therefore the next command.

The individual commands

In this section, the purpose of the commands and their function is further explained.

The OPEN command: the CLOS command

The OPEN command causes the program to set up an area on disk ready for use as a bank file. For this reason, operand 1 is valid, and may hold one of the values 08, 10, 12, or 14. Second operands are irrelevant.

Table 2

Restrictions on first operands for OPEN, FILE, and DUMP

Operand	Valid	Comment
01	No	Program utility work space: may not be referenced.
02	Yes	(FILE or DUMP) Tape unit, or sequential disk data.
03	No	Standard printer reference.
04	Yes	As for 02.
05	Yes	(FILE only) Standard card reader reference.
06	No	Standard printer reference.
07	No	Standard card punch reference.
08	Yes	1st bank disk file (random access).
09	No	Dictionary for 1st bank file: direct reference invalid.
10	Yes	As for 08, but 2nd file.
11	No	As for 09, but 2nd file.
12	Yes	As for 08, but 3rd file.
13	No	As for 09, but 3rd file.
14	Yes	As for 08, but 4th file.
15	No	As for 09, but 4th file.
16	Yes	As for 02.
Higher	No	Program restriction.

Table 3

FILE/DUMP: Operands and their interpretation

Op 1	Op 2	Comment
05		FILE only: a bank file must be open ready for use.
02	nnn	Read/Dump card images from/to the specified logical unit starting at data set nnn. A bank file must be open.
04	nnn	
16	nnn	
08		Read/Dump the dictionary associated with the specified bank file into/from store. May be used to open a file before use.
10		
12		
14		

Only one of the logical units 08, 10, 12, or 14 may be 'open' at a time, so that if an OPEN command is given for a new file, while the old file is still open, the old file is first automatically closed using the DUMP command (see below), which causes the old dictionary to be stored for future use before the new dictionary is prepared. After the OPEN command, the new file is assumed to be empty. However, there are two possible conditions of the empty file. When a direct access file is first used, it is 'preformatted' by the support software: a file made ready by the 'OPEN' command may thus be classed as 'created' or 'not-created'. This distinction is important when the FILE command is used to open a file (see FILE command below), for a file which has not been created, can have no dictionary. To prevent failure in this situation, a 'creation-flag' is used to control the commands which can be executed on a file.

The CLOS command informs the program that a given bank file is no longer required. All pointers and counters which control file access are cancelled, and any information still in the file becomes irretrievable unless the dictionary has previously been saved. Use of a CLOS command immediately before an OPEN command bypasses the automatic DUMP operation which is executed on the old file before opening the new file. Since only one bank file can be open at any time, only the current file can be acted upon by the CLOS command; operands are therefore unnecessary. After a CLOS command, no bank transactions can take place until a new file has been opened.

The FILE command: the DUMP command

The FILE command allows data to be inserted to a bank file. Alternatively, it permits the reading of a dictionary into store, so that a previously created bank file may be accessed. All valid operands shown in **Table 2** are permitted, and further details are shown in **Table 3.**

The DUMP command reverses the action of the FILE command. The value 05 for the first operand is not valid, but all other values may be used (see Tables 2 and 3). When data is dumped to units 02, 04, and 16, the current bank file remains open for further transactions, but when the operand has the value 08, 10, 12, or 14, the bank file is closed after use as if a CLOS command had been subsequently given. Since only the current bank file can be used in a dump, the use of 08, 10, 12, or 14, is merely a check that the correct file is present. If this check is not required, then omission of the operands for the DUMP command forces a dictionary dump for the current file, whatever its reference number.

The EDIT command

No operands are required since only the current bank file may be used. Editing information which follows an EDIT command may comprise

(i) 'STATus' cards,
(ii) 'DATA' cards, and
(iii) 'DLET' cards.

For a STATus card, the status information for the question is retrieved and checked. If column 6 on the STATus card contains a 1, any discrepancy is altered in favour of the card.

For a DATA card, the information is added to the other data cards associated with the question. If the only DATA information with the question is a dummy DATA card formed because the original question had no actual data, then the new DATA card overrides this dummy.

For a DLET card, the question reference is deleted from the dictionary, and the space on the bank file is released.

The three types of edit card may be input in any order, and in any quantity.

The DICT command: the WRIT command

These commands dump details of the current bank file. The DICT command dumps the contents of the dictionary, and also provides a check for duplicated references. The WRIT command dumps the entire contents of the current bank file itself. This includes all text, and data relating to the questions. Due to the quantity of output that this may involve, it is clear that only small files should be dumped using this facility.

The WORK command

No operands are required. Within the service routine, WORK data cards are read (as distinct from WORK command cards, which are read by the supervisor program). These have a content shown in **Table 4,** and they are similar in content to DATA cards contained within questions.

Cards containing a question reference amount to a specific request for that question via a dictionary search. All cards not containing a question reference amount to a general request for any question whose data matches the various headings on the WORK request card. Each entry is optional, although sufficient should be entered to avoid producing too many questions.

The KWIC command

Like the WORK command, the KWIC command is

Table 4

Parameters on a WORK data card

Subject heading
Examination reference
Four statistical parameters
Two alias names
A question reference

used to make selections. This command permits the insertion of a textual profile which forms the basis of the request. The profile is inserted on KWIC data cards (analogous to WORK data cards). If column 6 on the card is non-blank, then the text area (columns 7–72) is taken as a direct continuation of the text area on the previous card. Within the text area, or contiguous text areas, the profile is inserted as character strings enclosed in quotation marks ('). Several strings may be inserted in a single text area; each string is enclosed in quotes, and separated from other strings by at least one card column. The text from a question must contain at least one of the strings from a KWIC text area (or contiguous text areas) to be selected (OR procedure). In addition, several independent text areas may be set up, simply by inserting more request cards with text areas which are not continuations of previous cards. Each new text area may be termed a 'line', although a line may occupy more than one card. In this respect, the format is similar to standard FORTRAN program cards. A question must contain at least one of the strings (OR procedure) from every line (AND procedure) to be selected. An example of a KWIC profile request compiled from KWIC data cards is shown in **Table 5:** this might be expected to result in the selection of questions concerning traffic-lights (*sic*).

The STOP command

This command terminates the job with suitable close down procedures. It is used as the last card in the chain of commands.

Example of the use of the BANK program

Table 6 shows an example of a data stream for the BANK program. The complete data stream is composed of commands intermixed with other data as required.

In the example, a file exists (has been defined in the J.C.L. for the job step) on unit 08: the dictionary for this is read (from unit 09) enabling the file to be accessed. A question called ABC123 has an error, and a new version is to be substituted: the current version is first edited out (EDIT—DLET—EDIT), and the new version is then added (FILE—question—FILE). A dictionary listing is requested: this will be for the whole file. The WORK cards request the current (i.e. new) version of ABC123 to be listed, after which the file is no longer needed and is to be dumped to tape (unit 02, file 1).

A new file is then requested (OPEN). Since the previous file is still 'open', and only one file may be open at a time, the program will close file 08 by giving its own DUMP command. Note that this will conveniently save the dictionary for file 08 so that the disk file may be used at a later date (this does not affect the tape copy which is also available as backup). Some questions are read into file 10, and these are dumped to another tape (until 02, file 2). The facilities of J.C.L. enable this second tape data set to be located independently of the tape data set used in the first DUMP command, even to the extent of locating it on a different volume. The only exception is that the second data set cannot be both on the same volume, and physically in front of the first, otherwise the first data set will clearly be lost.

Following this second DUMP, the job ends. Since the current file (10) is still open when the STOP command is given, the file is first closed with a dictionary dump, as happened with the first file (08). This interlocking facility of the commands controlling opening, closing, and reading/dumping is extremely useful, for it reduces the number of command cards needed, and removes some of the risk of inadvertently losing data. However, the multiple functions remain logically consistent, and should not lead to confusion.

Table 5

Example of a KWIC request matrix listing

.AND.	LINE 1		'YELLOW'
		.OR.	'AMBER'
.AND.	LINE 2		'RED'
.AND.	LINE 3		'GREEN'

Table 6

Example of a BANK program data stream

Data Stream			Comment
FILE 08			Open a file and read dictionary.
EDIT			Commence edit.
DLET		ABC123	Remove question ABC123 from file.
EDIT			End of edit.
FILE 05			Read new questions from cards.
TEXT	(text)	ABC123	New
.	.	.	version
.	.	.	of
STAT 5	10100	ABC123	question
DATA	(data)	ABC123	ABC123.
FILE			End of questions.
DICT			Print dictionary of current file.
WORK			Commence selections.
WORK		ABC123	Find and write question ABC123.
WORK			End of selections.
DUMP	02001		Dump current file to tape.
OPEN 10			Open new file.
FILE 05			Read questions from cards.
	(questions)		
FILE			End of questions.
DUMP	02002		Dump to another tape.
STOP			End.

Discussion

A considerable number of university teachers, chiefly in faculties of medicine, are now taking advantage of the various scoring and analysis procedures offered for multiple choice questions. A country-wide research organisation is run, on essentially informal lines, from the Middlesex Hospital Medical School, and this is primarily aimed at medical faculties. The upsurge of in-course assessment, with its concomitant advantages has demanded some form of automation, and multiple choice questions (M.C.Q.) have proved useful in this field. This should not be taken to exclude other forms of assessment, but merely to place them in their proper perspective.

In solving most of the technical, administrative, economic, and ethical problems associated with M.C.Q. examination scoring and analysis, the area relating to question paper storage and preparation has been largely neglected: but many of the system constraints apply equally to scoring, and to question storage. In particular, the (centralised) computer facilities must be capable of simple presentation and interface with the teacher or examiner, so that he need not concern himself with actual computer operations.

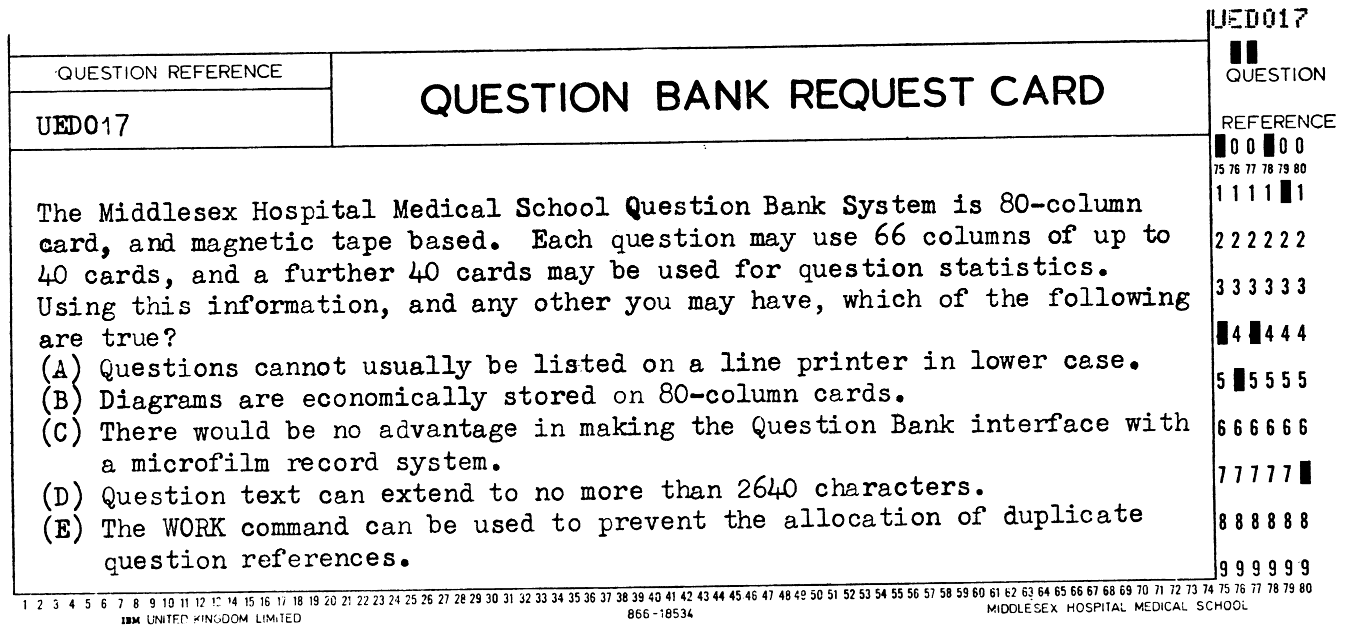
Whatever system is devised, it must be economical. While computer manufacturers (notably), and others, can spend large sums of money on complex systems, programming, and hardware, they will make little progress in the general university market as long as most university departments are forced to think in terms of an education budget of a few pounds, rather than hundreds, or thousands of pounds. This may be for a very long time indeed.

With the program written, it appears that it might be suitable for other applications. A parallel field of study has been the maintenance of student records on file, with name and other free text information coupled to course details and test results. It now appears that this is a good example of a similar system which could be similarly treated. No doubt, other examples will come to light given time.

As with every other system, a computer program is but a small part, and QBANK3B is supported by a well defined protocol. While there has not been space to consider this in detail, one item is of interest. A computer card has been designed (see **Fig. 1**) to enable the originator of the questions, and others if desired, to gain rapid access to the Bank file. The question may be typed onto this card, or a copy of the listing of the question may be fixed to the card: the latter having the advantage that the stored (and reproducible) version is shown, while the former has the advantage that the card will still pass through normal unit-record equipment.

With the current volume of information, data maintenance would be impossible without computer assistance. However, at any level of operation, the system must balance effectiveness against cost. While the University College London Computer Centre has been extremely helpful throughout the project (and indeed, without them this system would not be in existence), they suffer from a lack of peripherals, and the program wastes a considerable amount of processor time due to its requirements for input/output. At the moment, a batch processing procedure is adequate, and the cost of computer time is conveniently defrayed by the University computer services. As the quantity of data enlarges, batch processing will become less suitable. There is no doubt that the potential of the system can only be fully exploited when it has been implemented on a local dedicated computer, which needs only medium speed, but large random access storage. Such a computer should also be capable of connection to a more powerful machine for 'number-crunching', where needed. The use of on-line terminals to such a machine would also be required as the active participation of the user will enable decisions on data movement to be taken as problems arise, rather than at present, where a complex job must be planned, in detail, in advance. This is a situation which, of itself, limits the maximum usable size of the Bank and its files of data.

In spite of these difficulties, it is felt that the command language approach has been fully justified. With such a language, modules of program may be written independently (to a certain extent) to deal with each command, and these can be varied and updated with



UED017

QUESTION REFERENCE

UED017

QUESTION BANK REQUEST CARD

QUESTION REFERENCE

```
The Middlesex Hospital Medical School Question Bank System is 80-column
card, and magnetic tape based. Each question may use 66 columns of up to
40 cards, and a further 40 cards may be used for question statistics.
Using this information, and any other you may have, which of the following
are true?
(A) Questions cannot usually be listed on a line printer in lower case.
(B) Diagrams are economically stored on 80-column cards.
(C) There would be no advantage in making the Question Bank interface with
    a microfilm record system.
(D) Question text can extend to no more than 2640 characters.
(E) The WORK command can be used to prevent the allocation of duplicate
    question references.
```

IBM UNITED KINGDOM LIMITED 866-18534 MIDDLESEX HOSPITAL MEDICAL SCHOOL

Fig. 1 Sample question on computer card

little interference from other parts of the program. The supervisor which knits the modules together, is also capable of rapid modification. It is therefore proposed to extend this experience by writing other command interpreter programs for other similar projects.

If the reader can now answer the question in Fig. 1, the letters of the correct choices will be found in the third word of this sentence.

Acknowledgements

The authors would like to thank the staff of University College London Computer Centre for their assistance. Mrs. D. Evans has been largely responsible for the recording of the Bank of nearly 6000 multiple choice questions onto cards. The authors are indebted to Dr. P. Samet, Director of the UCL Computer Centre for his advice in the preparation of this manuscript.

References

Anderson, J., Wood, H., and Tomlinson, R. W. S. (1968). *Brit. J. Med. Educ.*, Vol. 2, p. 210.

Buckley-Sharp, M. D., and Harris, F. T. C. (1970). The Banking of Multiple Choice Questions, *Brit. J. Med. Educ.* (Vol. 4, p. 42). Bulletin 11: An Examiner's Guide to SCORE Version 7L (released to users).

Groves, P. D. (1968). *The Computer Journal*, Vol. 10, p. 365.

Harris, F. T. C., and Buckley-Sharp, M. D. (1968). *Brit. J. Med. Educ.*, Vol. 2, p. 48.

International Business Machines (U.K.) Ltd. Form C28-6539.

Book Review

Artificial Intelligence through Simulated Evolution, by Lawrence J. Fogel, Alvin J. Owens, and Michael J. Walsh, 1966; 170+xii pages. (*New York, London, Sydney: John Wiley and Sons Ltd., £3.75*)

An ancient principle in the design of machines is to adopt or to adapt methods used by nature. Although obvious, the principle is important and often overlooked so that it merits a name, say the *Naturist Principle*. It is usually necessary to bring two or more ideas together: birds do not fly by rotating their beaks nor by jet propulsion. An aeroplane is a cross between a bird and a sycamore seed or perhaps a squid.

The naturist principle can be applied to the study of machine intelligence by trying to copy the tricks of language, of the nervous system, and of the evolution of intelligence including the principles of natural selection and mutation. A fair amount of work has been done using the first two of these three approaches, and the third (evolutionary) approach is also not entirely new; it was, for example, suggested, by Oliver Selfridge in the 1958 symposium on the mechanisation of thought processes at the National Physical Laboratory. But the present book gives an account of what were perhaps the first fairly extensive experiments based on the idea. The idea should of course not be confused with the machine simulation of evolution for the study of evolution itself. (See, for example, J. L. Crosby, *New Scientist*, 21st February 1963.)

The individual 'machines' simulated in the experiments are all small finite-state automata. The simulation on a general-purpose computer is almost essential for the experiments owing to the continual redesign of the automata. The tasks put to these automata are the prediction of the next elements in sequences of letters, and sometimes there is an element of control as well. The more successful automata are allowed to give birth to new automata, with slight modifications including additions. Some measure of success is achieved for prediction problems that are simple enough, for example, when the original sequence is periodic and when the problem has a simple approximate solution. Practical implications are not yet evident since evolutionary techniques are not necessary for such simple problems. Moreover the bare description of the experiments makes it difficult to see the wood for the trees. But a beginning has been made.

The experiments are relevant to the status of simplicity in the mechanical or mental construction of concepts since, in some of the experiments, automata were handicapped in accordance with their complexities. Any given simple hypothesis is more likely to be approximately true, and also, for a variety of methods of concept generation, more likely to be generated than any given more complicated hypothesis: *this is why life is possible*. This would usually be true irrespective of the precise definition of 'simplicity' and irrespective of the method of hypothesis generation, be it by linguistic transformation, by pseudorandom artificial or real neural networks (the human method) or by simulation of evolution. Once an approximately correct hypothesis is generated, then it will tend to be confirmed (and hence consolidated in an adaptive technique) in virtue of its correct predictions. The present work, partly in virtue of its title, will help to channel research in these directions.

The automata that occur in the experiments are much simpler than unicellular animalcules, so it is appropriate that sexual reproduction has not yet been simulated. But eventually it will need to be since it would give scope to the combination of the good features of both 'parents'. This would be a natural strategy for the design of a creative machine, since, as Arthur Koestler has emphasised, creativity always involves the bringing together of two distinct ideas.

In addition to the description of the experiments there is also some speculative discussion. When discussing hard science the style is cold and dry, but it gets warmer when the science gets softer. Examples of the two styles are: (i) 'It is also evident that the cost matrix which expresses the goal of comparison must embody the characteristics that are the basis for human judgments of similarity' and (ii) '. . . the scientific method was not invented, it was discovered. It existed long before man; in fact it gave rise to man. Natural evolution can be looked upon as a realisation of the scientific method'. (Cf. the quotation heading Chapter 1 of Warren Weaver's 'Lady Luck'.)

There is no name index. For this the publishers are more to blame than the authors, since the production of a name index is a routine job which the publishers should organise after the page proofs are available.

I. J. Good (Oxford and Blacksburg, Virginia)

[*Editor's Note:* There was an unfortunate delay in the preparation of this review for publication, which was in no way the reviewer's responsibility. We apologise to the authors for the delay.]