DECUS PROGRAM LIBRARY

DECUS NO.	8-394
TITLE	BASIC MOO
AUTHOR	Guy L. Steele, Jr.
COMPANY	410 Washington Street Brighton, Massachusetts
DATE	March 3, 1971
SOURCE LANGUAGE	BASIC

BASIC MOO

DECUS Program Library Write-up DECUS NO. 8-394

CONTENTS

WARNING AND DISCLAIMER!!!!!

The game of MOO is highly addictive; use with caution! Many people become so fascinated that they will sit at a teletype for hours on end, often consuming many dollars' worth of computer time in a single sitting. Also, when not actually playing, they tend to become preoccupied thinking about ways to improve their play, and thus appear to withdraw from the real world. The author must therefore disclaim all responsibility for the consequences of anyone's learning to play Moo.

I. INTRODUCTION

MOO is a highly complex game of deductive strategy, requiring precise logical analysis; it is also a game simple enough to be elegant and easy to learn.

MOO was (to the best of the author's knowledge) originally implemented on the MULTICS timesharing system at the Massachusetts Institute of Technology. The MULTICS MOO system is writtin in PL/I; it not only plays the game, but keeps a table of all players' average scores and also their average times while playing.

MOO spread, and was also implemented in the APL language on MIT's CP/CMS timesharing system. The author became intrigued with the game on this system, and decided to popularize the game with his friends and other people. The author has written MOO in the FORTRAN and BASIC languages (the latter of which is the version presented here), and may soon write one in FOCAL if someone doesn't beat him to it.

This documentation is divided into seven sections. The first you're reading right now. The second consists of the Official Rules of MOO as obtained from the MIT CP/CMS version. The third section is an attempt to clarify the contents of the second section. The fourth is a listing of the BASIC MOO program and a sample run. The fifth section is a detailed explanation of the program's inner workings and contains flow charts. The sixth is a detailed explanation of the type of logic used in playing the MOO game. The seventh contains miscellaneous data about the MOO game.

One last word: don't be discouraged if the game seems difficult at first, because it is! But, once you get the idea of it, it is a really fascinating game. Also remember that although experts at MIT and elsewhere play the game, less advanced players thoroughly enjoy the game too. One fourth-grade teacher had arranged for a tour of a computer facility, and in preparation explained the game of MOO to her class; when the children went on the tour they were allowed to play MOO, with about eight of them collaborating on a single game, and they played a pretty good game! (In other words, if <u>they</u> can do it, so can you.) Good luck and have fun!

II. THE OFFICIAL RULES OF MOO

A game of Moo is initiated by the computer which selects (at random) four unique numbers in the range Ø to 9, inclusive. The player is then given as many tries as he desires to determine both the identity and the order of these four numbers.

At each turn within a game, the computer will ask you to type four numbers. For the first trial only of a game, you may type -1 as the first number if you desire to quit. At all other times the computer will continue to prompt you until you type four numbers. The computer will then compare the four numbers you have typed with the four numbers it has chosen and give you some feedback as follows:

(1) For each number you have typed which is both an element of the set of four numbers chosen by the computer, and in the position chosen by the computer, you are awarded a Bull (indicated by the appearance of a "B" in the computer's response).

(2) For each number you have typed which is an element of the set of four numbers chosen by the computer, but is not in the correct position, you are awarded a Cow (indicated by the appearance of a "C" in the computer's response).

(3) The computer tells you only how many Bulls and Cows are awarded, but not which numbers caused you to get them.

When you have guessed (or deduced) the exact sequence of four numbers, you are awarded four Bulls, and the game terminates with the computer telling you how many tries it took you and what your average for the current invocation of MOO is. A new game is then initiated.

III. HOW TO UNDERSTAND "THE OFFICIAL RULES OF MOO"

Unfortunately, the rules of MOO as given in section II are not the easiest in the world to understand. This section is intended to illustrate some of the main points.

At the beginning of each game, four digits are selected at random; these digits are all different (this is important). For example, the computer might choose 4, 7, 6, 8 or 1, 9, 5, 2 but not 1, 1, 3, 4 or 6, 7, 3, 7.

Suppose that the computer chooses 5, 2, 6, 8. Now suppose that the player guesses 5, Ø, 2, 6. The computer compares the two sequences and comes up with the following information:

1. The player's 5 matches the computer's 5 in the correct position. For this a Bull is awarded.

2. The player's Ø does not match any of the computer's digits. Nothing is awarded.

3. The player's 2 matches the computer's 2, but not in the correct position. For this a cow is awarded.

4. The player's 6 matches the computer's 6, but not in the correct position. For this a cow is awarded.

Thus, for this guess the player is awarded a total of 1 Bull and 2 Cows; the computer responds by typing "B, CC."

Now suppose that for the player's second guess he types 4, 2, 6, 6. (Notice that the player is not restricted to typing his digits all different; this is also very important.) The computer comes up with:

1. The player's 4 matches nothing. No award.

2. The player's 2 matches, in the right position, Bull.

3. The player's first 6 matches, in the right position, Bull.

4. The player's second 6 matches too, but not in the right position, Cow.

Thus, the computer responds with "BB, C."

For his third guess, suppose the player types 5, 6, 2, 8. The computer would then reply "BB,CC." (Can you see why?)

If, for his fourth guess, the player typed 4, 0, 0, 4, the computer would not award either Bulls or Cows, and would respond simple ",".

In summary, the following points should be made clear:

*The computer chooses four numbers which are all different.

*The player need not type in four different numbers on each guess.

*Each number typed in by the player is compared to each number chosen by the computer. If a given player's digit matches any of the computer's digits, it will win a Cow if its position differs from the computer's, or a Bull if its position is the same.

*If two or more of the player's digits are the same, either all of them will win something or none will. Each one will receive an individual award. However, no more than one of the matching digits can possibly win a Bull, i.e., if the player guesses 5, 5, 4, 4, he can get no more than 2 Bulls, 1 for a 5 and 1 for a 4. He may get fewer than 2 Bulls, and if one 5 wins a Bull, the other must win a Cow, and if one 5 wins a Cow, the other must win a Cow or a Bull; similarly for the 4's.

A discussion of deductive strategies will be found in Section VI.

IV. PROGRAM LISTING AND SAMPLE RUN

```
1∅ REM *** GAME OF MOO ***
11 REM
12 REM    GUY L. STEELE, 410 WASHINGTON ST., BRIGHTON, MASS
13 REM
2∅ REM *** PROGRAM INITIALIZATION
7∅ DIM N(9),R(3)
8∅ RANDOMIZE
9∅ LET G=∅
91 LET A=∅
92 FOR I=∅ TO 9
93 LET N(I)=I
94 NEXT I
99 REM *** BEGIN A NEW GAME - SET UP 4 NUMBERS IN N(∅) TO N(3)
1∅∅ LET T=∅
115 FOR I=∅ TO 3
12∅ LET K=INT(I+RND(∅)*(1∅-I))
125 LET X=N(I)
13∅ LET N(I)=N(K)
135 LET N(K)=X
14∅ NEXT I
149 REM *** READ PLAYER'S GUESS
15∅ PRINT
152 PRINT "TYPE 4 NUMBERS";
155 INPUT R(∅),R(1),R(2),R(3)
156 REM *** BRANCH IF HE WANTS TO QUIT
157 IF  R(∅) = -1  GOTO 9∅∅
159 REM *** BRANCH TO 8∅∅ IF ANY NUMBER INPUT IS BAD
16∅ FOR I=∅ TO 3
163 IF  R(I) < ∅  GOTO 8∅∅
165 IF  R(I) > 9  GOTO 8∅∅
167 IF  R(I) <> INT(R(I))  GOTO 8∅∅
17∅ NEXT I
171 REM *** ALL NUMBERS ARE OK - STEP COUNT OF TRIES
173 LET T=T+1
174 REM *** CALCULATE BULLS AND COWS
175 LET B=∅
177 LET C=∅
18∅ FOR I=∅ TO 3
182 FOR J=∅ TO 3
185 IF  R(I) = N(J)  GOTO 2∅∅
187 NEXT J
19∅ GOTO 27∅
2∅∅ IF  I = J  GOTO 25∅
21∅ LET C=C+1
22∅ GOTO 27∅
25∅ LET B=B+1
27∅ NEXT I
275 REM *** PRINT OUT BULLS AND/OR COWS
28∅ FOR I=1 TO B
29∅ PRINT "B";
3∅∅ NEXT I
31∅ PRINT ",";
```

```
32Ø FOR I=1 TO C
33Ø PRINT "C";
34Ø NEXT I
345 REM *** UNLESS HE GOT 4 BULLS, GO TRY AGAIN
35Ø IF B < 4  GOTO 15Ø
355 REM *** CALCULATE AND PRINT PLAYER INFO, THEN START NEW GAME
36Ø LET G=G+1
37Ø LET A=A+T
375 PRINT
38Ø PRINT "GAME";G;": TRIES =";T;", AVERAGE =";A/G
39Ø PRINT
4ØØ GOTO 1ØØ
79Ø REM *** PLAYER TYPED AN ILLEGAL NUMBER - BAWL HIM OUT AND RETRY
8ØØ PRINT
81Ø PRINT "NUMBERS SHOULD BE INTEGERS IN THE RANGE Ø TO 9."
82Ø GOTO 15Ø
89Ø REM *** PLAYER WANTS OUT - LET HIM ONLY AT THE START OF A GAME
9ØØ IF  T = Ø  GOTO 999
9Ø2 REM *** OTHERWISE SAY HE IS NAUGHTY AND MAKE HIM FINISH THE GAME
9Ø5 PRINT
91Ø PRINT "IT IS ANTISOCIAL TO TRY TO QUIT IN THE MIDDLE"
92Ø PRINT "OF A GAME OF MOO. PLEASE CONTINUE."
94Ø GOTO 15Ø
999 END
```

/NOTE: ALL INPUT TYPED/
/BY THE PLAYER IS /
/UNDERSCORED IN THIS /
/LISTING. /

```
RUN

TYPE 4 NUMBERS? Ø,Ø,1,2
,CCC
TYPE 4 NUMBERS? 1,Ø,Ø,1
,CC
TYPE 4 NUMBERS? 3,4,5,5
,
TYPE 4 NUMBERS? 6,7,8,8
,CCC
TYPE 4 NUMBERS? 8,7,7,8
B,C
TYPE 4 NUMBERS? 8,6,2,Ø
BB,CC
TYPE 4 NUMBERS? 8,2,6,Ø
BBBB,
GAME 1 : TRIES = 7 , AVERAGE = 7


TYPE 4 NUMBERS? Ø,Ø,1,2
,CCC
TYPE 4 NUMBERS? 1,Ø,Ø,1
,CC
TYPE 4 NUMBERS? 3,4,5,5
B,CC
TYPE 4 NUMBERS? 4,4,5,4
BB,CC
TYPE 4 NUMBERS? 4,2,5,Ø
BBBB,
GAME 2 : TRIES = 5 , AVERAGE = 6


TYPE 4 NUMBERS? Ø,1,2,3
,CC
TYPE 4 NUMBERS? 3.14159,6,9,3

NUMBERS SHOULD BE INTEGERS IN THE RANGE Ø TO 9.

TYPE 4 NUMBERS? 2,-2,5,6

NUMBERS SHOULD BE INTEGERS IN THE RANGE Ø TO 9.

TYPE 4 NUMBERS? 1,1,Ø,1
BB,CC
TYPE 4 NUMBERS? -1,Ø,Ø,Ø

IT IS ANTI SOCIAL TO TRY TO QUIT IN THE MIDDLE
OF A GAME OF MOO. PLEASE CONTINUE.

TYPE 4 NUMBERS? 3,4,4,5
,
TYPE 4 NUMBERS? 6,7,7,8
,CCC
TYPE 4 NUMBERS? 7,6,6,6
B,CCC
TYPE 4 NUMBERS? 1,6,Ø,7
BBBB,
GAME 3 : TRIES = 6 , AVERAGE = 6

TYPE 4 NUMBERS? -1,Ø,Ø,Ø

READY
```

AN EXPLANATION OF THE METHODS USED TO DEDUCE THE NUMBERS IN THE ABOVE GAMES MAY BE FOUND IN SECTION VI.

V. EXPLANATION OF PROGRAM LISTING AND FLOW CHART

The following is a detailed explanation of the inner workings of the MOO program. It is provided to help the user to modify the program if desired. (Suggestions: install an option to print out a brief set of rules, add coding to keep track of scores over an extended period of time for several players, so that tournaments can be held.) It should also be useful to those who just want to know exactly how it works.

VARIABLE USAGE

N(9) contains in its ten elements N(Ø) to N(9) the ten digits Ø to 9. The contents of N are scrambled randomly before each game; the four numbers which then are in elements N(Ø) to N(3) are the four digits chosen by the computer for that game.

R(3) contains in the elements R(Ø) to R(3) the four numbers typed in by the player at each turn.

G contains the total number of games played so far.

A contains the total number of tries made by the player in all previous games. A player's average is computed by dividing A by G.

T contains the number of tries made by the player so far in the current game. After each game A in increased by T, and T is then reset to Ø.

B contains the number of Bulls awarded on a given try; used to control printing.

C contains the number of Cows awarded on a given try; used to control printing.

I used in FOR loops and for indexing.

J used in FOR loops and for indexing.

K a randomly computed index used for scrambling N.

X a temporary variable used for exchanging two elements of N when scrambling.

PROGRAM OPERATION

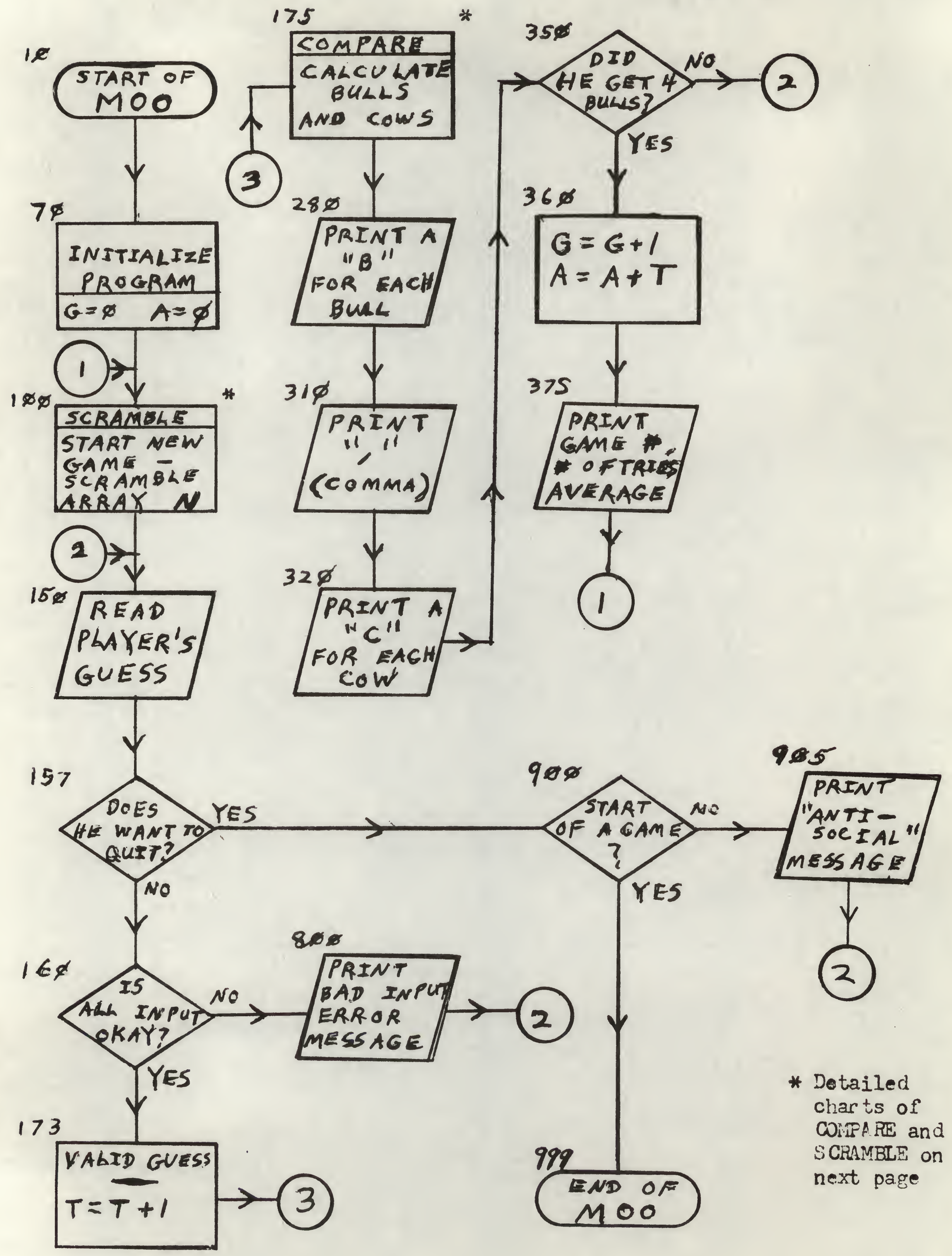
Lines	Function
1Ø-13	Heading and credits.
2Ø-94	Initialize program. Dimension N(9), R(3); randomize the random number generator; set number of games G and total tries A to Ø; set the elements of N to the digits Ø to 9.
99-14Ø	Begin a new game. Set number of tries so far T to Ø; scramble the elements of N. The latter is done by choosing at random any of the ten elements and exchanging it with N(Ø); then choosing any of N(1) to N(9) and exchanging it with N(1); then choosing any of N(2) to N(9) and exchanging it with N(2):

	then choosing any orN(3) to N(9) and exchanging it with N(3). This results in having four random digits in N(Ø) to N(3), all different. Since the values were exchanged and not merely replaced, the entire array N still contains all ten digits for use on the next scramble.
149-155	Read player's guess. Print message "TYPE 4 NUMBERS" and input four digits into R(Ø) to R(3).
156-157	Branch if quit desired. If the first number typed by the player was -1, branch to line 9ØØ to check the validity of the request.
159-17Ø	Check the validity of the numbers typed in. Each of the four numbers typed by the player in R(Ø) to R(3) are checked; a branch is made to line 8ØØ if any of them is not an integer in the range Ø-9.
171-173	All numbers are okay. The current try by the player is valid; add 1 to the number of tries for the current game T. (This is not done until after data checking so that the player is not penalized for typing errors.)
174-27Ø	Calculate Bull/Cow awards for the current try. Bet B and C to Ø; then use a double FOR loop to compare each number typed by the player to each one chosen by the computer. Whenever a match is found, then the indexes I and J of the FOR loops are compared; if they match, a Bull is awarded (1 is added to B); if not, a Cow is awarded (1 is added to C). Note that the combined values of B and C may never exceed 4.
275-34Ø	Print out Bulls and Cows. Print out the number of "B's" specified by B; then print ", "; then the number of "C's" specified by C.
345-35Ø	Check for end of game. Branch to line 15Ø for another try unless the previous try yielded 4 Bulls (i.e., the player determined all 4 digits and their positions.)
355-4ØØ	Print out info to player. Add one to number of games G; add number of tries used into A; print the number of the game just completed, the number of tries used for that game, and the average number of tries over all previous games. Branch to line 1ØØ to start a new game.
79Ø-82Ø	Player typed illegal input. Print an error message, then branch to line 15Ø to retry. The player is not penalized for the faulty entry.
89Ø-9ØØ	Check validity of quit request. The player is only allowed to quit at the start of a new game; thus, a branch to the END at line 999 is made only if T is Ø, i.e. no tries were previously made for the current game.
9Ø2-94Ø	Yell at player for trying to quit. If the player tried to quit in the middle of a game, a message is printed saying he is antisocial and to please continue. (The individual user may want to replace this message with one of his own choice.) Then a branch is made to line 15Ø to retry. The player is not penalized here either for being naughty; a penalty may be enforced by adding a

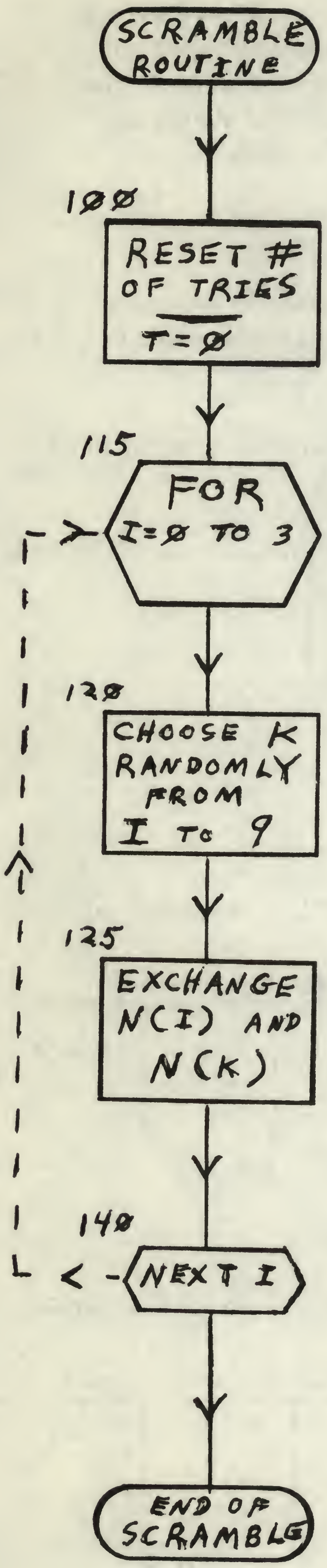
line of the form 93Ø LET T=T+1 which costs the player a try.

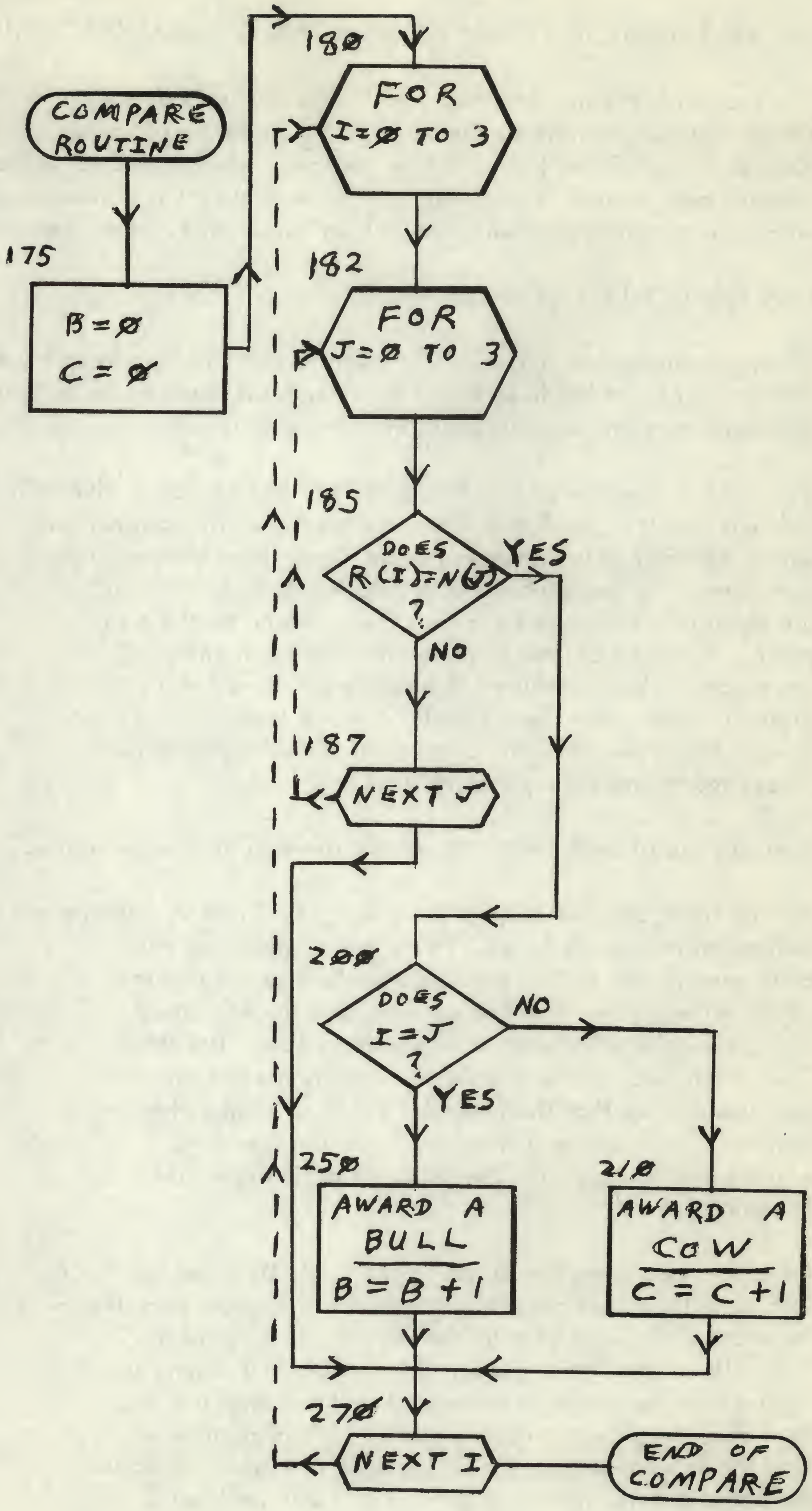
999 End of program.

FLOWCHART   Note: the numbers to the upper left of each box are line numbers.



* Detailed charts of COMPARE and SCRAMBLE on next page


SCRAMBLE ROUTINE
100
RESET # OF TRIES
T=0
115
FOR I=0 TO 3
120
CHOOSE K RANDOMLY FROM I TO 9
125
EXCHANGE N(I) AND N(K)
140
NEXT I
END OF SCRAMBLE


COMPARE ROUTINE
175
B=0
C=0
180
FOR I=0 TO 3
182
FOR J=0 TO 3
185
DOES R(I)=N(J) ?
YES
NO
187
NEXT J
200
DOES I=J ?
NO
YES
250
AWARD A BULL
B=B+1
210
AWARD A COW
C=C+1
270
NEXT I
END OF COMPARE

VI. EXPLANATION OF SAMPLE RUN AND GENERAL DISCUSSION OF MOO STRATEGY

This section is divided into two parts. The first part demonstrates the strategy used during the sample run (which, by the way, is not an optimal strategy, just a workable one) and methods by which the Bull and Cow feedback may be used to deduce rather than guess the identity and position of each digit. The second part is a discussion of various simple and not-so-simple strategies which have been developed and of their relative merit.

EXPLANATION OF SAMPLE RUN

Whenever the author plays MOO he uses a chart to help keep track of possible solutions. This type of chart has been found by other MOO hackers to be helpful also; therefore it is described here and used in explaining the sample run.

The chart is sometimes called a "pi-chart" because of its shape; it is basically a set of three columns (fig 1). The first column is used for writing down digits which are definitely known not to be along those chosen by the computer. The second column is used for those digits which are definitely known to be among those chosen by the computer. The third column has one entry for each entry in the second column, indicating possible positions for the digits in column 2. These possibilities are later narrowed down. The area above the crossbar of the pi is also sometimes used for temporary scratch work.

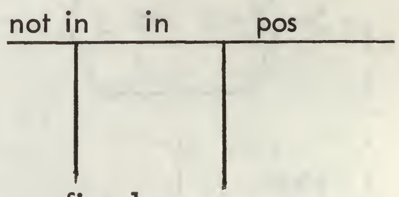


fig. 1

With the aid of such charts the sample run will now be explained.

For the first guess the player typed Ø, Ø, 1, 2 and in response got ",CCC." A number of things can now be deduced. First, two of the Cows must have been gotten by the zeros, since the 1 and 2 together could not win more than two awards, and since if one Ø won a Cow the other must win something also. The third Cow, then, was won either by the 1 or by the 2. We can also deduce that the computer's Ø must be in either position 3 or 4, since if it were in position 1 or 2 we would have won a Bull. The result so far is summarized in figure 2.

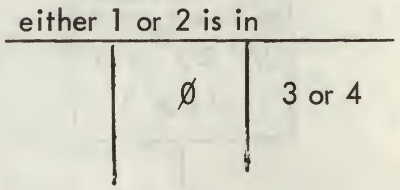


fig. 2

For his second guess the player typed 1, Ø, Ø, 1 and got ",CC." Since the Ø's won something last guess they must have won something this guess also; therefore both Cows were won by the Ø's and none by the 1's. This tells us that the computer did not have a 1 among its digits; from the previous guess we thus know that the 2 must be among the computer's digits. We also know that the Ø is not in position 3, because if it were we would have won a Bull, since this guess had a Ø in position 3. Therefore the Ø must be in position 4. (Fig. 3)

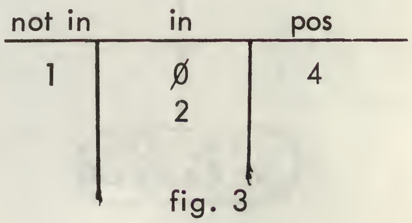


fig. 3

For his third guess the player typed 3, 4, 5, 5 and got ",", i.e. no awards. This tells us that of the digits 3, 4, and 5, none is among the computer's digits. (fig. 4).

not in	in	pos
1 3	Ø	4
4 5	2	

fig. 4

For his fourth guess the player typed 6, 7, 8, 8 and got ",CCC." By the reasoning used for the Ø's earlier, two of the three Cows were won by the 8's, and one Cow by either the 6 or the 7. We know then that the four digits chosen by the computer are Ø, 2, 8, and either 6 or 7; so we can immediately eliminate 9. We can also say that the 8 is in position 1 or 2, since we would have gotten a Bull otherwise. (fig. 5)

either 6 or 7 is in

1 3	Ø	4
4 5	2	
9	8	1 or 2

fig. 5

For his fifth guess the player typed 8, 7, 7, 8 and got "B, C." Since the 8's won awards last guess, the Bull and Cow belong to the 8's this guess; the 7's won nothing, so 7 is out and 6 is in. Also, since one of the 8's won the Bull, the computer's 8 must be in position 1 or 4 (the positions of the 8's in the guess typed in); since the Ø is in position 4, the 8 is in position 1. (fig. 6).

not in	in	pos
1 3	Ø	4
4 5	2	
9 7	8	1
	6	

fig. 6

We can now see from the chart that there are only two possibilities left for the computer's digits; 8, 2, 6, Ø or 8, 6, 2, Ø. The player chose to guess the latter on his sixth try and got "BB, CC", and finally got four Bulls on his seventh guess with 8, 2, 6, Ø.

A game like this might faze some people, but this player is a compulsive MOO hacker, so he started right in on a second game.

For his first move the player typed Ø, Ø, 1, 2 and again got ",CCC." The same information derived during the first game applies here also; the Ø's got two of the Cows, and either the 1 or the 2 got the third Cow. Also, the Ø is in position 3 or 4. (fig. 7)

either 1 or 2 is in

	Ø	3 or 4

fig. 7

For his second move the player again typed 1, Ø, Ø, 1 and again got ",CC" (really unusual; such coincidences are due to the random number generator and fortunately don't happen often). Anyway, the information on the last game's second move also applies here: the Ø is in position 4; the 2 is in; the 1 is out. (fig. 8)

not in	in	pos
1	Ø	4
	2	

fig. 8

For his third guess the player typed 3, 4, 5, 5 and this time got "B, CC". Now there are a number of possibilities; we know that the two 5's must have won two of the three awards, but we don't know whether they won two Cows or the Bull and one Cow. Also, whatever wasn't won by the 5's must be assigned to the 3 or the 4.

3 in pos 1 and 5 in pos 2
or 5 in pos 1 and 4 in pos 2
or 3 in pos 2 and 5 in pos 3
or 4 in pos 1 and 5 in pos 3

If the 5's did win two Cows, then the 3 or the 4 won the Bull; and the computer's 5 must be in position 1 or 2 (or else we would have won another Bull for one of the 5's typed in). If the 3 won the Bull, then the 3 is in position 1, and the 5 in position 2; if the 4 won the Bull, we have the 4 in position 2 and the 5 in position 1.

1 6	Ø	4
7 8	2	
9		

fig. 9

Now suppose that the 5's won a Bull and a Cow. Then we have the computer's 5 located in position 3 or 4; since we already have the Ø placed in position 4, the 5 would be in position 3. Now if the 3 had won the Cow, the computer's 3 could not be in position 1 (or else it would have won a Bull), nor in position 3 or 4 (since those are occupied); similarly, if the 4 had won the Cow, it would have to be in position 1.

Finally, since all of the above possibilities account for all of the computer's digits, 6, 7, 8, 9 can't be among them and are therefore out. (fig. 9)

A long and complex string of deductions, but this is the challenge of MOO. And, the effort is worth it; we'll have the answer by the fifth guess!

For his fourth guess the player typed 4, 4, 5, 4 and got "BB, CC." Since no more than one of the 4's could have won a Bull, the 5 must have won the other one. We may therefore say that the computer's 5 is in position 3, thus eliminating the first two alternatives in figure 9. Since the 4's of this guess did together win two Cows and a Bull, there must be a 4 among the computer's numbers; this rules out the third alternative in figure 9, so the fourth alternative must be the correct one. (fig. 10).

not in	in	pos
1 6	Ø	4
7 8	2	
9 3	4	1
	5	3

fig. 10

It is easily seen from figure 10 that the 2 must by default be in position 2; therefore for his fifth guess the player typed in 4, 2, 5, Ø and received four Bulls.

The third game will be left as an exercise for the reader (but don't feel compelled to go through all the reasoning as above; instead, play your own games of MOO; you'll learn more by experimenting).

GENERAL DISCUSSION OF MOO STRATEGY

There are three ways that one may play MOO; or rather, there are three goals one may play for. The beginner plays for the most obvious goal: on each game to eventually get four Bulls. The intermediate player is more concerned with lowering his average number of tries so that he can get his four Bulls faster (also so he can boast about his low MOO average!). The veteran MOO hacker is a player who is obsessed with finding the optimal MOO strategy, so he can lower his average, so he can get four Bulls faster. This third type of player actually doesn't spend much time playing MOO; he does spend a lot of time daydreaming about MOO at work or in class! (See warning at the beginning of this documentation.)

In any event, the work done by these veteran Moo players does provide valuable advice to less experienced MOO players. The results so far are discussed here.

* *

The simplest approach is to begin by typing four identical digits: Ø, Ø, Ø, Ø or 1, 1, 1, 1 for example. After no more than nine tries this method will give you the identity of the four numbers chosen by the computer. The positions can then be located by choosing a number which is known not to be in the set chosen by the computer (assume it is Ø) and one which is in the set (say 4) and then typing Ø, Ø, 4, 4 and Ø, 4, Ø, 4. The following possible combinations of responses will then indicate the position of the 4:

Response to Ø, Ø, 4, 4,	Response to Ø, 4, Ø, 4	Position of the computer's 4
B,C	B,C	4
B,C	,CC	3
,CC	B,C	2
,CC	,CC	1

This method will give the solution in about 12 to 15 tries. Not good!

A more complex approach is to begin by typing Ø, Ø, 1, 1. The second guess is determined by the response to the first:

,	2,2,3,3	
B,C	1,Ø, Ø, Ø	(response here of ,C, indicates a 1 in pos 3 or 4; response ,CCC indicates Ø in pos 1; B,CC indicates Ø in pos 2)
,CC	1, Ø, Ø, Ø	(response of ,C indicates 1 in pos 2; response B, indicates 1 in pos 1; response B,CC indicates Ø in pos 3 or 4)
,CCCC	1, Ø, Ø, Ø	(response of BB,CC indicates 1 in pos 1 and Ø in pos 3 or 4; response B,CCC indicates 1 in pos 2 and Ø in pos 3 or 4
B,CCC	1, Ø, Ø, Ø	(response of ,CCCC indicates Ø in pos 1 and 1 in pos 2; response BB,CC indicates 1 in pos 1 and Ø in pos 2; response B,CCC indicates either Ø in pos 3 and 1 in pos 4 or 1 in pos 3 and Ø in pos 4)
BB,CC	1, Ø, Ø, 1	(response ,CCCC indicates Ø in pos 1 and 1 in pos 3; response BB,CC indicates Ø in pos 2 and 1 in pos 4; response B,CCC indicates either Ø in pos 1 and 1 in pos 4 of Ø in pos 2 and 1 in pos 3)

A strategy using this type of approach on each guess will average about 9 tries per game.

A still more complex strategy involves using Ø, Ø, 1, 2 as the first guess, followed by 1, Ø, Ø, 1 unless the response was ",", in which case 3, 3, 4, 5 is used. When the position of at least one digit is known, it is believed best if typing a guess of the form a, b, c, c to arrange the digits so that one of the c's is in the position of the known digit; and if two digits and their positions are known, to arrange a guess a, b, c, c so that the a and one of the c's are in the positions occupied by the known digits. These arrangements have been empirically found to yield the most information; such generalizations, of course, are subject to modification depending on the particular game. This strategy, properly applied, generally yields an average of 6.

This is not the best, though! One MOO hacker at the Massachusetts Institute of Technology has played to date over 200 games, averaging 5.3 tries per game; several others average less than 5.5 tries per game. Their strategy is not completely understood by the author and will not be described here, except to say that on most guesses no duplicate digits are typed in.

In the final analysis, the best way to improve your MOO average is to play a lot of MOO yourself until you almost develop an "intuition" for the game.

VII. MISCELLANEOUS MOO DATE

VITAL STATISTICS ABOUT THIS IMPLEMENTATION

Programming language: BASIC.
Core requirements: Should easily fit on any BASIC system.
Execution time: Limited only by speed of teletype.
Goofability: Practically nil. All input is exhaustively checked for errors.
Hardware required: Any standard BASIC system, i.e. computer and teletype.

VITAL STATISTICS ABOUT MOO IN GENERAL

Best known average: 5.3 (over more than 200 games).
(Note: expert MOO hackers consider an average of 6 or over mediocre. An amateur does well to average 6 to 7.)
Shortest known game: 1 guess (he was lucky).
Longest known game: 37 guesses (he really didn't understand the game, presumably).
Number of ways the computer can choose four numbers: 5040.
Number of ways player can guess four numbers: 10000.
(Anyone for drawing a game tree?)
Maximum number of guesses per game: Infinite.

SUGGESTIONS TO OTHER PLAYERS WHO INDIVIDUALLY IMPROVE THEIR OWN IMPLEMENTATION OF MOO

Provide for the program to print out a short set of rules for MOO (if such a set can be written!)

Add a provision for automatically keeping track of several players' scores (this would require some device such as disk or DECtape, probably).

If your computer has a real-time clock, add a provision for timing games.

Put in an option for explaining to beginners why awards were made the way they were (this would imply letting the player peek at the answer - perhaps aid to a beginner could be given in some other way).

The author would appreciate any comments or suggestions on improving this MOO implementation or on improving MOO strategy.

<u>Addendum to DECUS number 8-394, BASIC MOO</u> June 26, 1972

BASIC MOO was originally developed for PDP-11 BASIC. This version of BASIC has the special feature that if in a FOR statement the initial value is greater than the final value, the entire loop is skipped (not executed). This is also true of PDP-10 BASIC.

This is <u>not</u> true of PDP-8 BASIC!!! In PDP-8 BASIC, such a FOR loop will be executed <u>once</u>.

The loops in statements 28Ø to 3ØØ and 32Ø to 34Ø depend on the feature of PDP-11 BASIC in the case when the variable B or C has the value Ø (zero). When B (C) is zero, the loop is not to be executed, so that no "B"s ("C"s) will be printed.

Thus the program will work correctly as written on a PDP-11 or PDP-10. On a PDP-8 the following two statements should be added to forestall the loops when B or C is zero:

```
278 IF B=Ø GOTO 31Ø
318 IF C=Ø GOTO 345
```

These will bypass the loops completely if B or C is zero.