STANFORD ARTIFICIAL INTELLIGENCE LABORATORY
MEMO AIM-225
STAN-CS-74-406

MEMORY MODEL FOR A ROBOT

BY

W. A. Perkins

SUPPORTED BY
ADVANCED RESEARCH PROJECTS AGENCY
ARPA ORDER NO. 2494
PROJECT CODE 3D30

JANUARY 1974

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

STANFORD ARTIFICIAL INTELLIGENCE LABORATORY
MEMO AIM-225

COMPUTER SCIENCE DEPARTMENT
REPORT NO. CS-406

JANUARY 1974

MEMORY MODEL FOR A ROBOT

by

W. A. Perkins*

Astract: A memory model for a robot has been designed and tested in a simple toy-block world for which it has shown clarity, efficiency, and generality. In a constrained pseudo-English one can ask the program to manipulate objects and query it about the present, past, and possible future states of its world. The program has a good understanding of its world and gives intelligent answers in reasonably good English. Past and hypothetical states of the world are handled by changing the state the world in an imaginary context. Procedures interrogate and modify two global databases, one which contains the present representation of the world and another which contains the past history of events, conversations, etc. The program has the abi l ity to create, destroy, and even resurrect objects in its world.

* Present address is General Motors Research Laboratories, Warren, Michigan 48090

This research was supported by the Advanced Research Projects Agency of the Department of Defense under Contract DAHC15-73-C-0435.

MEMORY MOOEL FOR A ROBOT

by

W. A. Perkins

TABLE OF CONTENTS

SECTION	PAGE
I. INTRODUCTION	2
II. DIALOG	6
III. DATABASES A. Knowledge Representation Database B. Historic Oat abase	46
IV. PROCEDURAL DESCRIPTIONS	50
	56
VI. CONTEXT MECHANISM	60
VII. DISCUSSION A. Self-debugging B. Generating Answers C. Reasoning 0. Creating, Destroying, and Resurrecting Tokens (Particular Objects) E. Suggestion for Future Work	63
VIII. ACKNOWLEOGMENTS	70
APPENDIX A. HISTORIC DATABASE GENERATE0 BY DIALOG	71
REFERENCES	115

I . INTRODUCTION

One of the main problems in Artificial Intelligence is the representation of knowledge. Early work on this important problem was done by McCarthy[1], Newell, Shaw, and Simon[2], Amarel[3], and McCarthy and Hays[4]. In natural language understanding Quillian[5], Winograd[6], and Shank[7] have spent a large part of their effort in this area. In computer vision the representation problem has been considered by Winston[8], Binford[9], and Agin and Binford[10]. New programming languages have been developed[11,12,13,14] to make representation of knowledge easier to work with. Several programs which attempt to model human memory from the psychological point of view are EPAM[15], SAL[16], ELINOR[17], and HAM[18].

One needs a system which can readily store, retrieve, and manipulate data. It must be flexible enough to work in a changing world and yet have enough depth to embody difficult concepts and relationships. Work in this direction has resulted in a program which can carry on a conversation about its world in a constrained language. No natural-language parsing is done. The input questions must satisfy a given format while the output is a little more natural. The memory-model system has been tested on Winograd's Block World[6] and it handled the questions with great speed and efficiency.

Figure 1 shows a possible segmentation for a robot system. Language input is via a teletype and visual input is via a TV camera with a "hand" for manipulation of objects. In Winograd' s system and the present system the "hand'" and servo-program are replaced with visual displays and the TV camera, the visual parser (Segments 5 and 6) and the procedures for storing and retrieving visual data (Segment 9) are all absent. The present system has been further constrained by elimination of the natural language parser (Segments 2 and 3) with a somewhat simpler executive (Segment 1). These simplifications allowed concentration on the robot' s memory model (Segments 4,7, and 8).

The program (MAX) is written in SAIL[19] which is an extension of ALGOL and has a good string structure. The program runs in about 100K of PDP10 core at a speed of about 10 times that of Winograd' s Program[6] (non-nat Ural-language part). The reasons for this gain in speed with the same computer is: (1) Difference in programming languages--SAIL versus LISP and MICROPLANNER; (2) Difference in database structure--string array retrieval versus pattern matching of database.

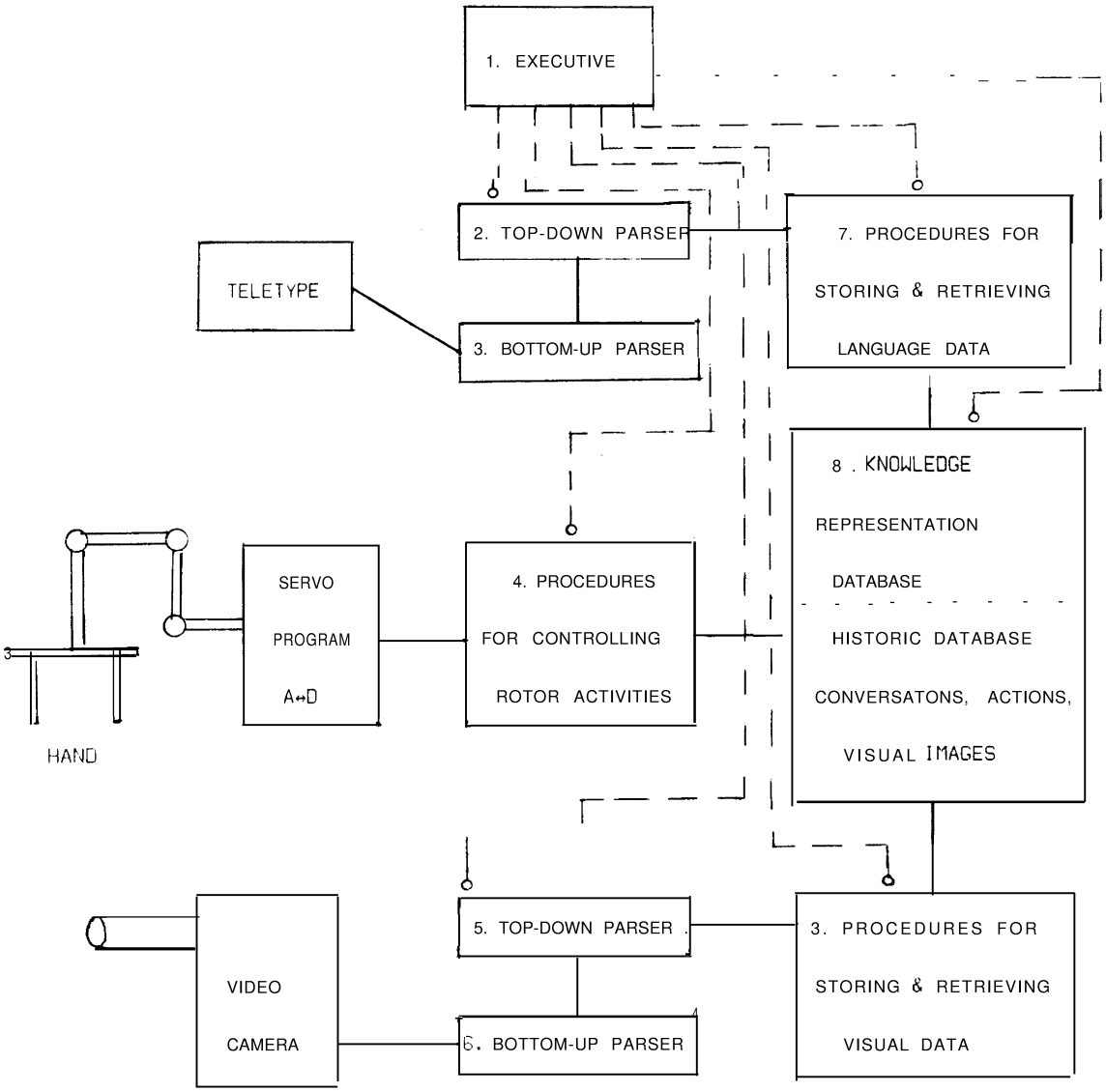

Figure 1. A possible segmentation of a robot.

The basic data storage unit for the program' s database is string arrays. Each location in the string array can be viewed as a storage location of unlimited length. By use of the SAIL scanning functions and a user-selected set of break characters, it is possible to quickly pick out the desired component from the string or list.

In its "static" database the program contains all the knowledge necessary to describe the world in its present state (see Knowledge Representation Database, Segment 8). Some of the knowledge about an object can be pulled directly out of an array such as the color, location, and name of objects that it supports, and names of objects that it is supported by. However, many descriptions involve manipulation of the database in order to retrieve the desired answer. For example, "Find a block which is taller than the one you are now holding" must be answered by manipulation of the database since it would be impractical to store on the property list of each block whether it was taller than the block now being held by the hand. Some forms of data such as color and size are ideal for storage on the property list while others such as "taller than . . . " are definitely not suitable. Other data such as support relationships and left-of relationships are in an intermediate category and could be handled either way. In the program we chose to add the support relationship as a property, but not left-of, right-of, etc. This means that as objects are moved, the program must continually update the properties top-status and bottom-status of all objects that are affected by the movements. However, the program can answer questions involving support with ease and speed. On the other hand the program does not have to waste time keeping track of the left-of relationship as it moves objects, but it must do a little calculation every time it wishes to know this relationship between two objects.

Segment 8 also contains historic data such as conversations and actions which have occurred. This information is stored in a linear (one index) array which we call the "Grapevine". Many procedures store information on this Grapevine and a smaller number interrogate it. Objects are readily described by reference to this Grapevine in such expressions as "the one I told you to pick up" and "objects that you touched while you were doing it".

Manipulation of blocks is done by a group of specialists. When the strategist or higher level procedure wants a particular Block A put on top of Block B, it does not have to concern itself with the fact that Block A is three deep in one pile and Block B is four deep in another pile and the

hand is holding some other object. It simply asks that the final job be done and each motion specialists calls on other specialists and sometimes itself recursively in order to set up the proper conditions for it to do its job. This is essentially the method used by Winograd[6,20] and we have found it very satisfactory.

. The program uses a context mechanism to answer questions about hypothetical and past states of the world. When asked if it can stack up some configuration of blocks, the program searches its past (Historic Database) to see if it has ever done such an operation. If not, it will enter an imaginary context and try to perform the operation. (The state of the real world is preserved.) The program can answer any question about the world in the past by entering the imaginary context, retracing its steps by use of information stored in the Historic Database, and then examine its database in light of this new context. There is no overhead for entering this imaginary context and the memory space increases are proportional to the size of the imaginary world (i. e. the extent of the modifications to the real world).

The program contains an extensive error checking system. Each procedure checks the syntax of its input. If a procedure fails, it stores the reason for failure on the Grapevine and reports failure to its caller. At present this failure usually propagates upward to the highest level and the program prints out on the teletype that it failed and the reason. It would be highly desirable to have higher-level procedures or the executive handle some of these problems, but this has only been done in a few special cases. However, this error-checking has turned out to be an excellent debugging technique as discussed in Section VII.

The program was written using the standard features of SAIL and does not use LEAP (see Ref. 19). It is interesting to note that the abilities of the program were achieved without using a high-level language such as PLANNER[11], QA4[12], CONNIVER[13], or MLISP[21]. It was possible to incorporate many of the ideas from those languages (such as procedural knowledge and context mechanism) into the program. The program was written over a period of eight months and represents about six-man months of effort (the listing is about 250 pages). One of the reason that it was possible to do this in such a short time is the fact that the program runs more rapidly than similar program in high-level languages and therefore new algorithms could be debugged in less time.

II. DIALOG

The following is a dialog between MAX (the program) and a friend, We have chosen Winograd's Block World[6,20] so this system could be compared with that important earlier work. Winograd chose his questions so that every second or third question would bring in some new problem for the representation system (a few of his questions were chosen only to test the natural language parser and are not very interesting in terms of the represent at ion) .

Besides testing our system on all of Winograd' s questions, we have added 22 new questions which test additional features of our system (these new questions are marked with a "&" on the right). The quest ions given to MAX are marked by a "#" and his answers follow the questions after a blank line. The natural language questions (in small letters) were inserted later. Following each answer we have inserted the PDP 10 run time (without displays) in square brackets. Comments are in curly brackets.

A deeper insight into what MAX was thinking about during this conversation can be obtained from Appendix A which will become understandable after reading Sections III, IV, V, and VI.

HELLO I' M MAX, WHAT CAN I DO FOR YOU?

{See Fig. 2. Some hidden lines in the back of objects are omitted or dotted, but an object is not obscured by another object in front of it. }

1. pick up a big red block.
YOU DO PICKUP BLOCK(COLOR = RED|SIZE = BIG).

OK
[run time = 0.4 sec]

{The display showed the hand grasping the big green cube, putting it on the table, and then picking up the big red block in going from Fig. 2 to Fig. 3. }

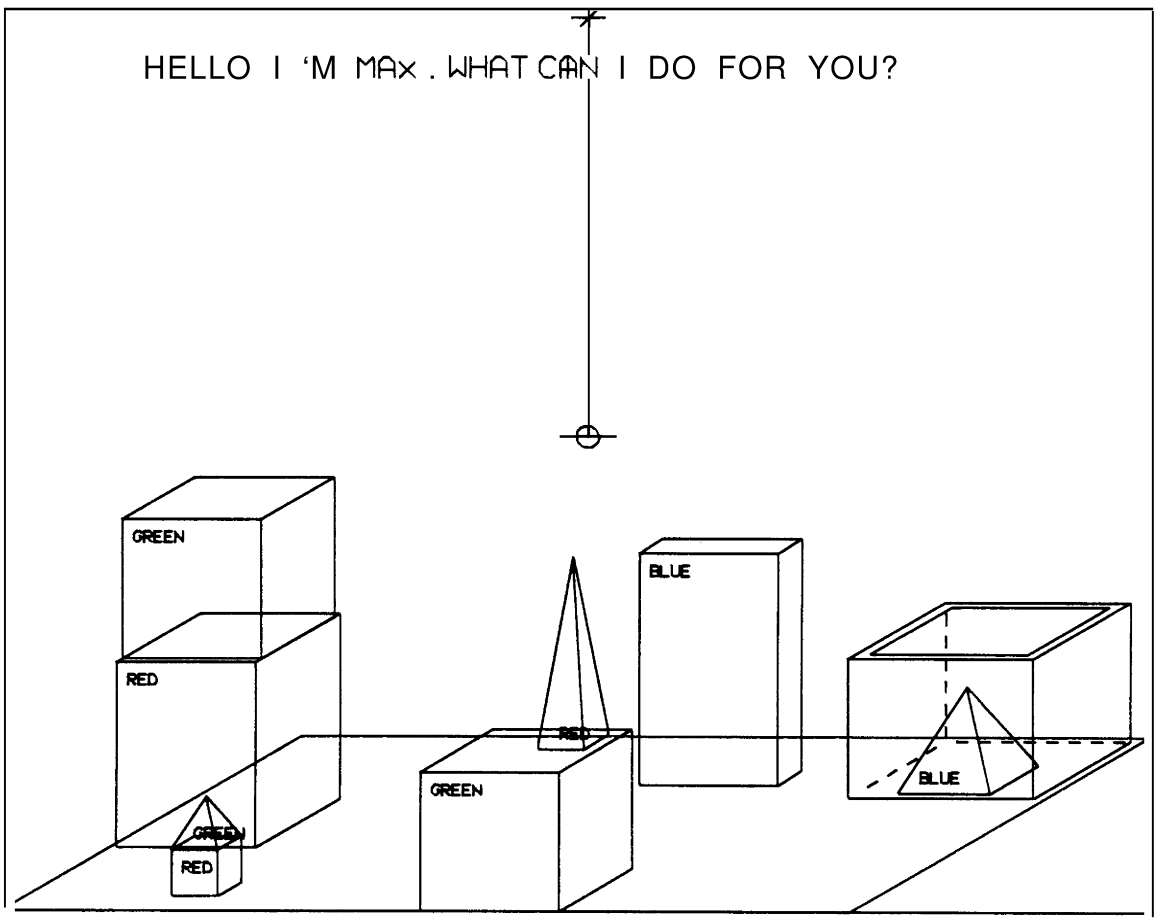
HELLO I 'M MAx . WHAT CAN I DO FOR YOU?
GREEN
RED
GREEN
RED
RED
GREEN
BLUE
BLUE

Figure 2.

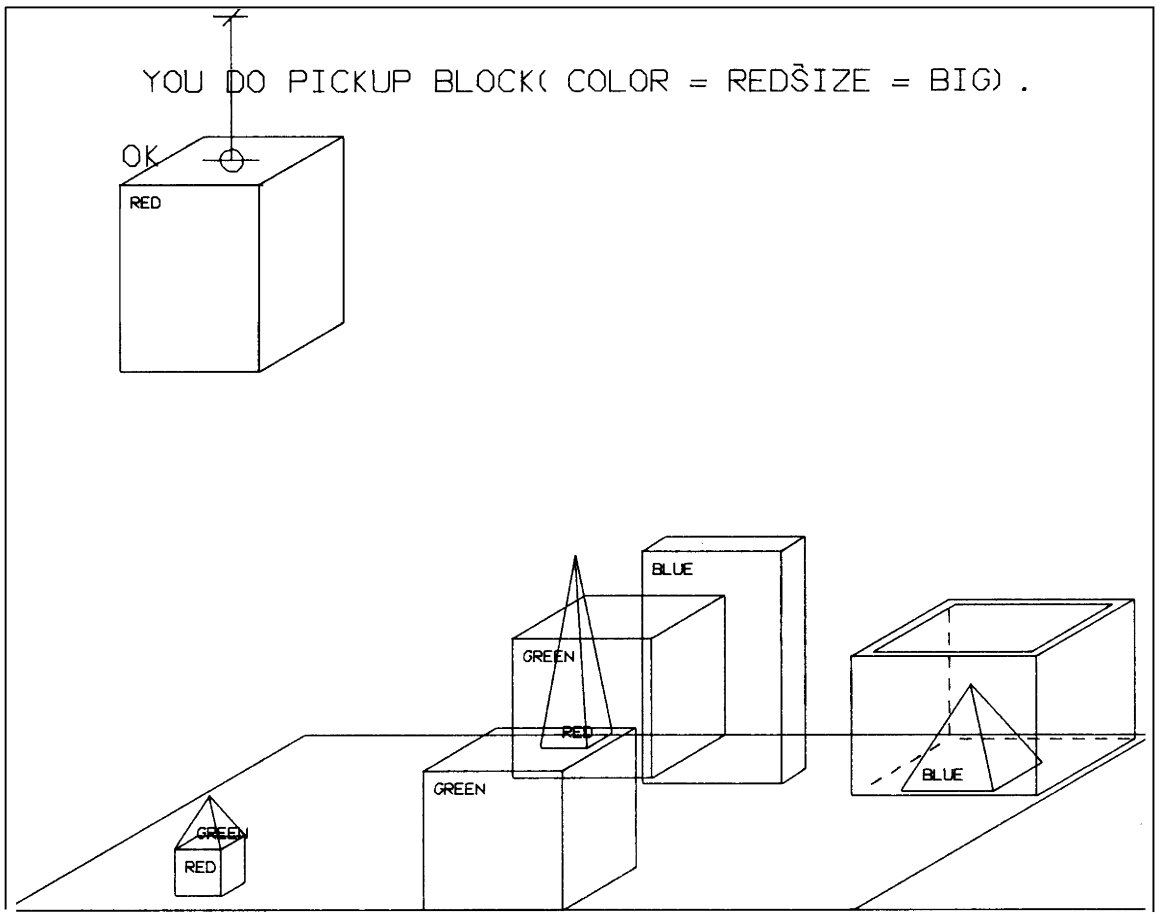
YOU DO PICKUP BLOCK(COLOR = REDŠIZE = BIG) .
OK
RED
BLUE
GREEN
RED
GREEN
BLUE
GREEN
RED

Figure 3.

2. grasp the pyramid.
YOU DO GRASP BLOCK(DEFINITE|KINDOF = PYRAMID).

COULDN' T DO IT BECAUSE DESCRIPTION FIT THE SMALL RED
PYRAMID THE SMALL GREEN PYRAMID THE SMALL BLUE PYRAMID
[run time = 0.3 sec]

{MAX was unable to carry out the operation because he found three objects which fitted the description, KINDOF = PYRAMID while DEFINITE indicated that he should find only one. }

3. find a block which is taller than the one you are holding
and put it into the box.
// FIND BLOCK(HEIGHT > HEIGHT[BLOCK(HELD_STATUS = GRASPED-BY
HAND(MAX))]) AND YOU DO BLOCK(ANS:↑FIND//X → LAST 2)
INSIDE BOX(DEFINITE) .

IT IS THE BIG BLUE PARALLELEPIPED
OK
[run time = 0.8 sec]

{Note that "it" was handled by reference to a previous answer. The hand put the big red block down and placed the big blue parallelepiped inside the box resulting in the scene of Fig. 4. }

4. what does the box contain?
FIND OBJECT(BOT,STATUS = ONTOP BOX(DEFINITE)).

IT IS THE BIG BLUE PARALLELEPIPED THAT. I AM NOW HOLDING
AND THE SMALL BLUE PYRAMID
[run time = 0. 4 sec]

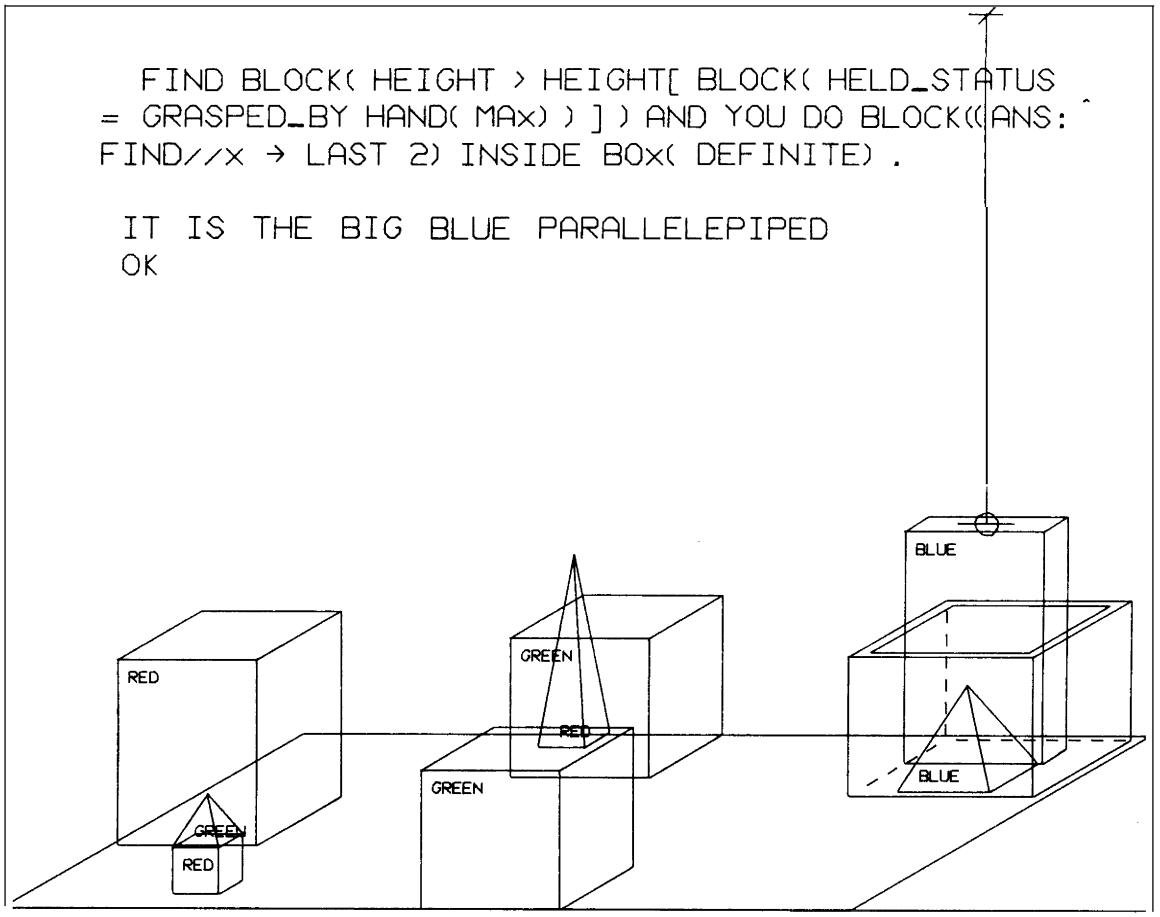
FIND BLOCK(HEIGHT > HEIGHT[BLOCK(HELD_STATUS
= GRASPED_BY HAND(MAx))]) AND YOU DO BLOCK((ANS:
FIND//x → LAST 2) INSIDE BOx(DEFINITE) .
IT IS THE BIG BLUE PARALLELEPIPED
OK
RED
GREEN
RED
GREEN
RED
GREEN
BLUE
BLUE

Figure 4.

5. what is the pyramid supported by?
FIND OBJECT(TOP_STATUS = HOLDING BLOCK(KINDOF = PYRAMID|
ANS:↑FIND//X ← LAST 2)).

IT IS THE BOX THAT I JUST PUT A BIG BLUE PARALLELEPIPED
INSIDE OF
[run time = 0. 7 sec]

'{Here the description in the command was essentially "the pyramid that you found" which was unique. }

6. how many blocks are not in the box?
HOW MANY BLOCK(BOT_STATUS ≠ ONTOP BOX(DEFINITE)|KINDOF
= PARALLELEPIPED)?

4 THE BIG RED PARALLELEPIPED AND THE BIG GREEN CUBE
WHICH IS TOWARD THE BACK AND THE BIG GREEN CUBE
THAT IS NOW HOLDING A SMALL RED PYRAMID AND THE
SMALL RED CUBE
[run time = 0.3 sec]

7. is at least one of them narrower than the one which i told you to pick up?
DOES EXIST BLOCK(ANS:↑EXIST/X/ ← LAST 2|YWIDTH <
YWIDTH[BLOCK(COM:HUMAN(FRIEND) TO HAND(MAX)↑PICKUP/
HAND(MAX)/X ← LAST ∞)])?

YES THE SMALL RED CUBE
[run time = 0.4 sec]

8. is it supported?
DOES EXIST OBJECT(TOP_STATUS = HOLDING BLOCK(ANS:↑EXIST/
X/ ← LAST 2))?

YES THE TABLE
[run time = 0. 6 sec]

9. can the table pick up blocks?
IS POSSIBLE TABLE DO PICKUP BLOCK()?

NO BECAUSE AN_VEG_MIN[TABLE(ITSELF)] NOT EQUAL ANIMATE
[run time = 0.02 sec]

(MAX was saying that since the property AN_VEG_MIN for the category TABLE was not ANIMATE, the answer was "no". }

10. can a pyramid be supported by a block?
// IS POSSIBLE BLOCK(KINDOF = PYRAMID) ONTOP BLOCK(KINDOF = PARALLELEPIPED)?

YES EXAMPLE IS THE SMALL RED PYRAMID AND THE BIG GREEN CUBE THAT IS NOW HOLDING A SMALL RED PYRAMID
[run time = 0.4 sec]

{In answering hypothetical questions such as this MAX first looks for an example. }

11. can a pyramid support a pyramid?
IS POSSIBLE BLOCK(KINDOF = PYRAMID) ONTOP BLOCK(KINDOF = PYRAMID)?

NOBECAUSE BLOCK(SHAPE_OF_TOP = POINTED) NOT SUPPORT ANY OBJECT
[run time = 0. 9 sec]

'{Since no example existed, MAX entered his imaginary world and tried to put a pyramid on top of a pyramid. During execution he discovered that this cannot be done and the reason. }

3. 2. could you stack up three red blocks?
Is POSSIBLE YOU DO STACKUP BLOCK(COLOR = RED) BLOCK(COLOR = RED) AND BLOCK(COLOR = RED)?

YES I IMAGINE I COULD DO THAT
[run time = 1.7 sec]

(MAX first checked to see if he had ever stacked up three red blocks. Since he had never done it, he entered his imaginary world and performed the action giving the result shown in Fig. 5.}

13. could you pick up a red block?
IS POSSIBLE YOU DO PICKUP BLOCK(COLOR = RED)?

YES I ALREADY DID THAT
[run time = 0.2 sec]

{In checking past actions, MAX discovered that he had already picked up a red block. }

14. could you stack up three blocks?
IS POSSIBLE YOU DO STACKUP BLOCK0 BLOCK0 AND BLOCKO?

YES I ALREADY FIGURED THAT OUT
[run time = 0.3 sec]

{Although MAX had not really stacked up three blocks, in his imaginary world he had done it and he remembered that, saving himself the effort of figuring it out again. }

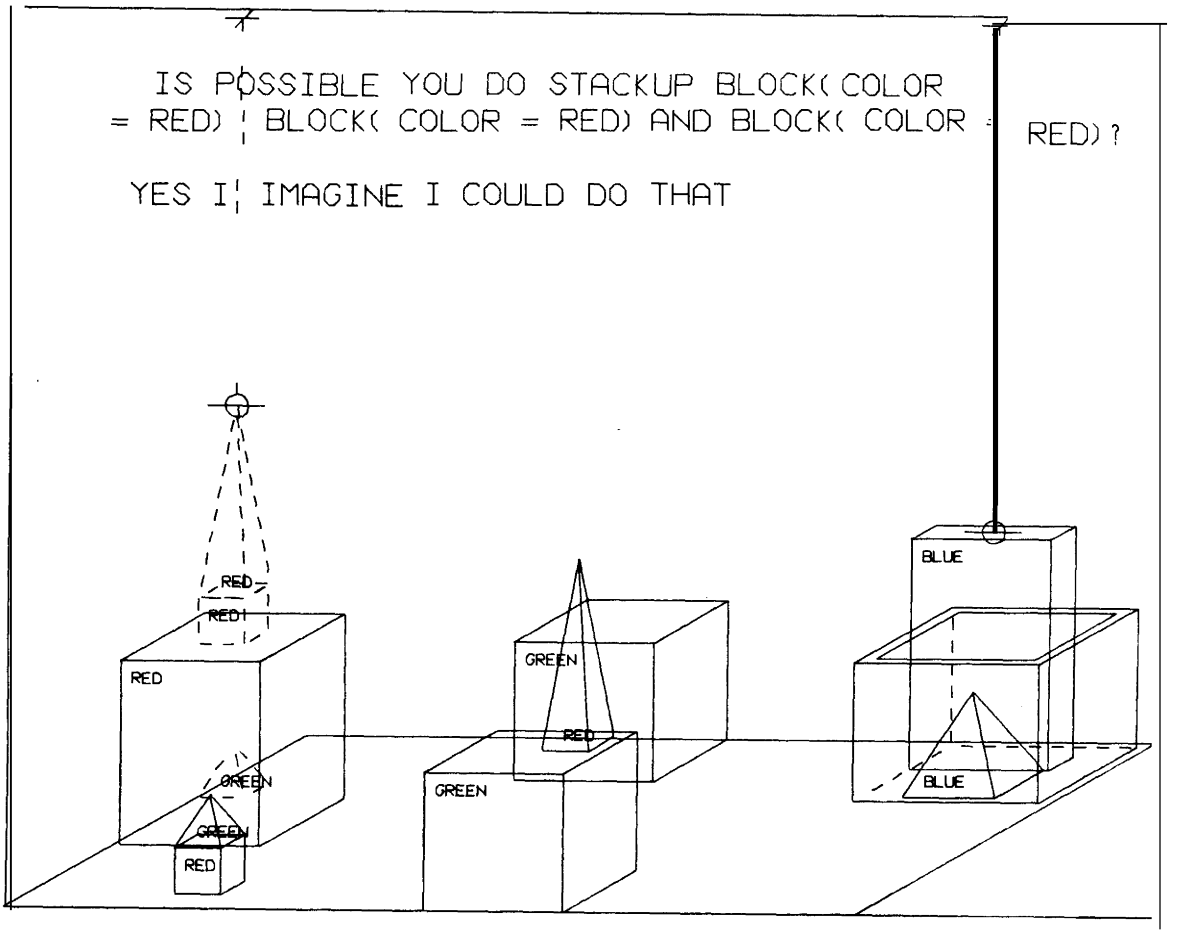
IS POSSIBLE YOU DO STACKUP BLOCK(COLOR = RED) ¦ BLOCK(COLOR = RED) AND BLOCK(COLOR = RED) ?
YES I¦ IMAGINE I COULD DO THAT
RED
RED
RED
GREEN
GREEN
RED
GREEN
RED
GREEN
BLUE
BLUE

Figure 5.

15. the blue pyramid is nice.
// STORE FACT LIKE-STATUS OF HUMAN(FRIEND) IS LIKE
BLOCK(DEFINITE|COLOR = BLUE|KINDOF = PYRAMID).

OK
[run time = 0.1 sec]

{Note how different the surface structure is from the form used to represent the meaning. }

16. i like blocks which are not red, but i don't like anything which supports a pyramid.
// STORE FACT LIKE-STATUS OF HUMAN(FRIEND) IS LIKE
BLOCK(COLOR ≠ RED) AND STORE FACT LIKE-STATUS OF
HUMAN(FRIEND) IS NOT LIKE OBJECT(TOP_STATUS = HOLDING
BLOCK(KINDOF = PYRAMID)).

OK
OK
[run time = 1 . 0 sec]

17. do i like the box?
// IS FACT LIKE-STATUS OF HUMAN(FRIEND) IS LIKE BOX(DEFINITE)?

NO BECAUSE LIKE_STATUS[HUMAN(FRIEND)] NOT EQUAL LIKE
BOX(DEFINITE)
[run time = 0.1 sec]

{MAX was only saying that there does not exist any fact that his friend likes the box. Therefore it is necessary to ask the next question.}

// IS FACT LIKE-STATUS OF HUMAN(FRIEND) IS NOT LIKE
BOX(DEFINITE)?

YES THE BOX
[run time = 0.1 sec]

18. do i like anything in the box?
DOES EXIST OBJECT(BOT_STATUS = ONTOP BOX(DEFINITE)|
LIKED-STATUS = LIKED-BY HUMAN(FRIEND))?

YES THE BIG BLUE PARALLELEPIPED THAT I AM NOW HOLDING
AND THE SMALL BLUE PYRAMID
[run time = 0.9 sec]

{Since (15) and (16) caused information to be stored on the property list of the objects as well as the human's property list, MAX can easily answer this quest ion. }

19. is the green cube which is in front of a green block liked or disliked by anyone?
WHAT IS LIKED-STATUS OF BLOCK(COLOR = GREEN|KINDOF = CUBE|
LOCASHUN IN-FRONT-OF BLOCK(COLOR = GREEN))?

NOT LIKED-BY HUMAN(FRIEND)
[run tirne = 0.2 sec]

{MAX was given contradictory information about his friend's feeling toward this block in (16). Evidently he decided to believe the most recent information. }

20. will you please stack up both of the red blocks and either a green cube or a pyramid?
// YOU DO STACKUP BLOCK(COLOR = RED) BLOCK(COLOR = RED)
AND EITHER BLOCK(COLOR = GREEN) OR BLOCK(KINDOF = PYRAMID).

OK
[run time = 0.9 sec]

(The configuration of Fig. 6 is obtained. }

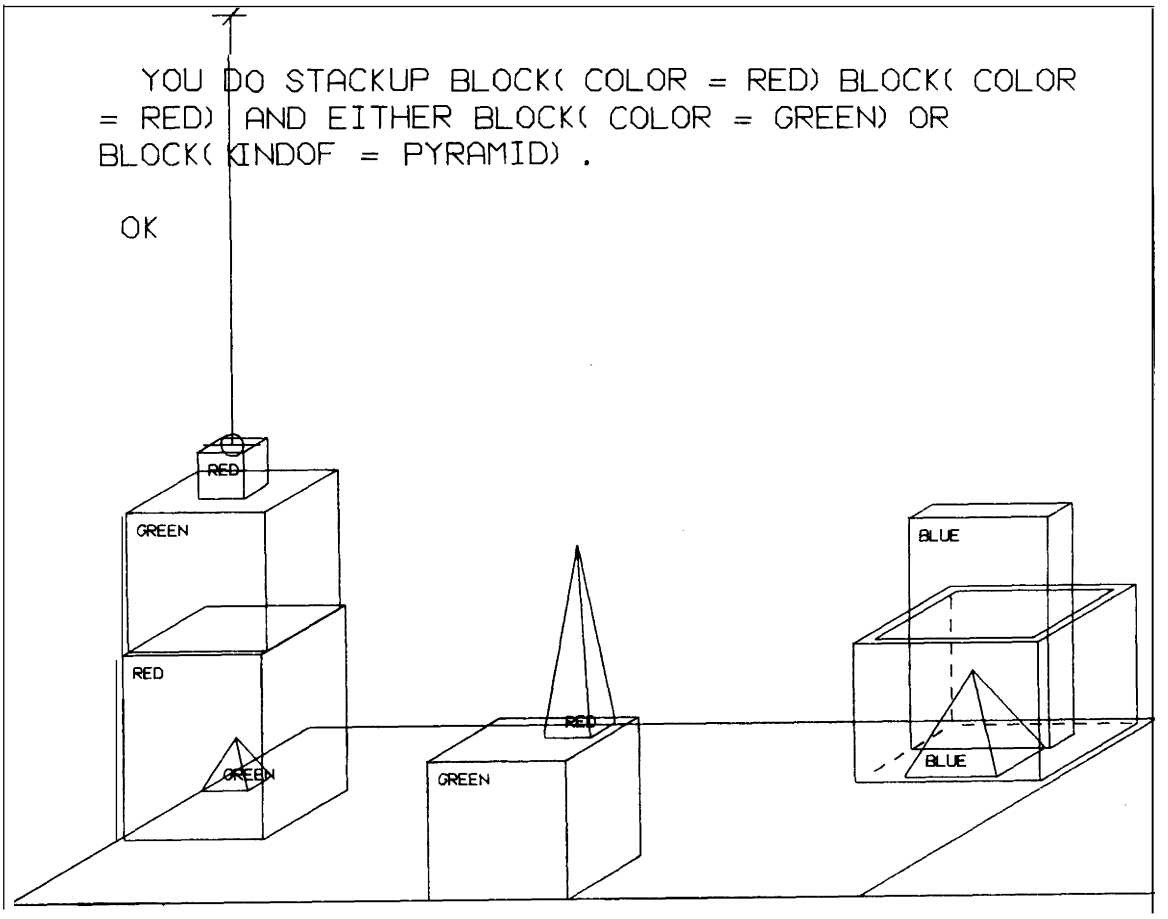
YOU DO STACKUP BLOCK(COLOR = RED) BLOCK(COLOR = RED) AND EITHER BLOCK(COLOR = GREEN) OR BLOCK(KINDOF = PYRAMID) .
OK
RED
GREEN
RED
GREEN
RED
GREEN
BLUE
BLUE

Figure 6.

21. which cube is sitting on the table?
FIND BLOCK(KINDOF = CUBE|BOT_STATUS = ONTOP TABLE(DEFINITE)).

IT IS THE BIG GREEN CUBE THAT IS NOW HOLDING A SMALL
RED PYRAMID
[run time = 0.3 sec]

22. what do the red pyramid and the small cube have in common?
WHAT DO BLOCK(COLOR = RED|KINDOF = PYRAMID) AND
BLOCK(KINDOF = CUBE|SIZE = SMALL) HAVE IN COMMON?

THEY BOTH BELONG TO SAME CATEGORY
THEY BOTH HAVE THE COLOR RED
THEY BOTH HAVE THE SIZE SMALL
THE LENGTH OF BOTH IS 50
THE WIDTH OF BOTH IS 50
THEY ARE BOTH ON TOP OF THE BIG GREEN CUBE'S
THEY ARE BOTH CLEAR ON TOP
[run time = 0.3 sec]

{Note that MAX not only looks at the similarities of the objects in question but checks the similarities of objects that they are on top of and support .}

23. what do the green cube which is holding a cube and the green cube which is holding a pyramid have in common?
WHAT DO BLOCK(COLOR = GREEN|KINDOF = CUBE|TOP_STATUS
= HOLDING BLOCK(KINDOF = CUBE)) AND BLOCK(COLOR = GREEN|
TOP-STATUS = HOLDING BLOCK(KINDOF = PYRAMID)|KINDOF
= CUBE) HAVE IN COMMON?

THEY BOTH HAVE THE DESCRIPTION THE BIG GREEN CUBE
THEY BOTH HAVE THE DIMENSIONS 150 150 150
THEY ARE BOTH HOLDING SMALL RED BLOCK'S

[run time = 0.6 sec]

24. is there a large block behind a pyramid?
DOES EXIST BLOCK(SIZE = BIG|KINDOF = PARALLELEPIPED|
XCOORD < XCOORD[∃ BLOCK(KINDOF = PYRAMID)])?

YES THE BIG BLUE PARALLELEPIPED AND THE BIG RED
PARALLELEPIPED AND THE BIG GREEN CUBE THAT IS NOW
HOLDING A SMALL RED CUBE
[run time = 0.5 sec]

25. put a small one onto the green cube which supports a pyramid.
YOU DO BLOCK(SIZE = SMALL|KINDOF = PARALLELEPIPED)
ONTOP BLOCK(COLOR = GREEN/TOP-STATUS = HOLDING BLOCK(KINDOF
= PYRAMID)).

OK
[run time = 0.4 sec]

{MAX moves the block and the scene of Fig. 7 results. }

26. put the littlest pyramid on top of it.
YOU DO BLOCK(KINDOF = PYRAMID|HEIGHT ≤ HEIGHT[∀
BLOCK(KINDOF = PYRAMID)]) ONTOP BLOCK(ACT:↑ONTOP/X/
← LAST 3).

OK
[run time = 8.6 sec]

{MAX moves the block and the scene of Fig. 8 results. }

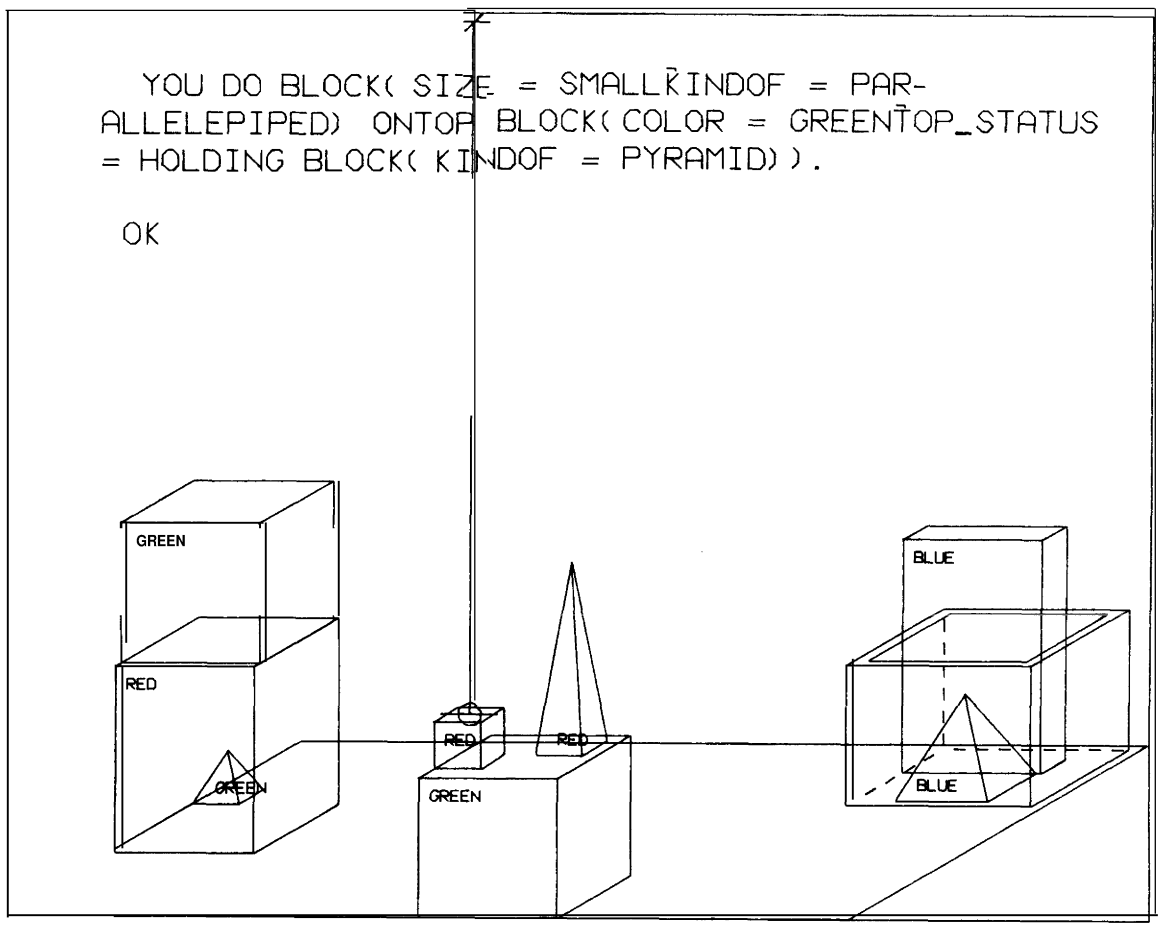
YOU DO BLOCK(SIZE = SMALL KINDOF = PAR-
ALLELEPIPED) ONTOP BLOCK(COLOR = GREEN TOP_STATUS
= HOLDING BLOCK(KINDOF = PYRAMID)).
OK
GREEN
RED
GREEN
RED
RED
GREEN
BLUE
BLUE

Figure 7.

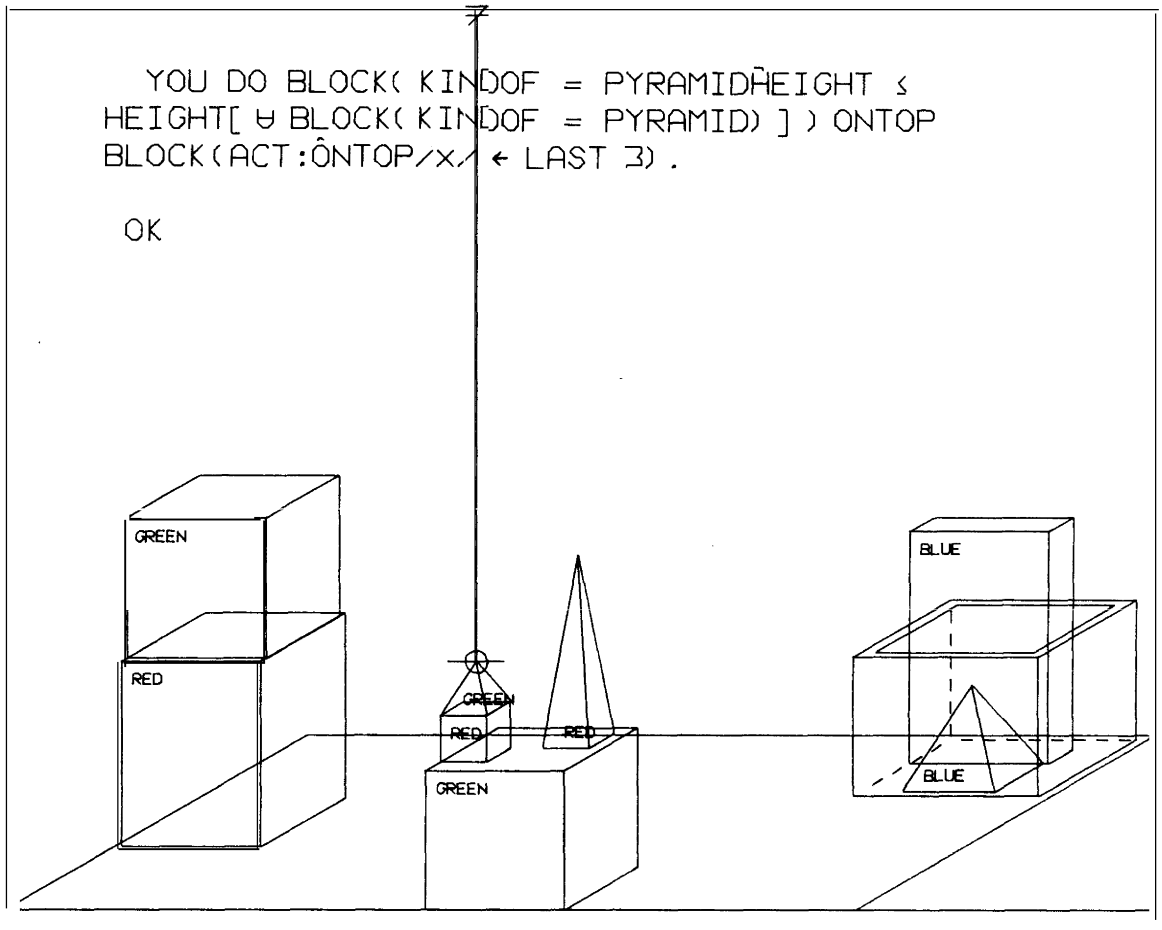
YOU DO BLOCK(KINDOF = PYRAMIDHEIGHT ≤
HEIGHT[∀ BLOCK(KINDOF = PYRAMID)]) ONTOP
BLOCK(ACT:ÔNTOP/X/ ← LAST 3) .
OK
GREEN
RED
GREEN
RED
RED
GREEN
BLUE
BLUE

Figure 8.

27. does the shortest thing the tallest pyramid' s support supports support anything green?

// DOES EXIST OBJECT(TOP_STATUS = HOLDING BLOCK(KINDOF = PYRAMID|HEIGHT ≥ HEIGHT[∀ BLOCK(KINDOF = PYRAMID)]))
AND DOES EXIST OBJECT(COLOR = GREEN|BOT_STATUS = ONTOP OBJECT(BOT_STATUS = ONTOP OBJECT(ANS:↑EXIST/X/ → LAST 3))|
HEIGHT ≤ HEIGHT[∀ OBJECT(BOT_STATUS = ONTOP OBJ ECT(ANS:↑ EXIST/X/ → LAST 3))])?

YES THE BIG GREEN CUBE THAT IS NOW HOLDING A SMALL RED CUBE AND A SMALL RED PYRAMID
YES THE SMALL GREEN PYRAMID THAT I AM NOW HOLDING
[run time = 7.5 sec]

// DOES EXIST OBJECT(COLOR = GREEN|BOT_STATUS = ONTOP OBJECT(HEIGHT ≤ HEIGHT[∀ OBJECT(BOT_STATUS = ONTOP OBJECT(TOP_STATUS = HOLDING BLOCK(KINDOF = PYRAMID|HEIGHT ≥ HEIGHT[∀ BLOCK(KINDOF = PYRAMID)])))]))?

YES THE SMALL GREEN PYRAMID THAT I AM NOW HOLDING
[run time = 18.7 sec]

{This is a difficult question to say the least. Since the question does not reveal to MAX that all of the objects are blocks and the fact that he does an exhaustive search makes this particularly difficult. Note that the first formulation with less nesting takes less time. }

28. what color is it?

// WHAT IS COLOR OF OBJECT(TOP_STATUS = HOLDING OBJECT(ANS:↑EXIST/X/ ← LAST 2))?

RED
[run time = 0. 5 sec]

{This question was chosen by Winograd because of the difficulty with "it" in the natural language part. }

29. how many things are on top of green cubes?
HOW MANY OBJECT(LOCASHUN DIRECTLY-ABOVE BLOCK(COLOR = GREEN|KINDOF = CUBE))?

3 THE SMALL RED CUBE AND THE SMALL RED PYRAMID
AND THE SMALL GREEN PYRAMID THAT I AM NOW HOLDING
[run time = 0.8 sec]

30. had you touched any pyramid before you put the green one on the little cube?
DOES EXIST BLOCK(KINDOF = PYRAMID|ACT:↑GRASP/HAND(MAX)/
X -- COM:↑ONTOP/BLOCK(COLOR = GREEN|KINDOF = PYRAMID)/
BLOCK(SIZE = SMALL|KINDOF = CUBE)- ∞)?

YES THE SMALL GREEN PYRAMID THAT I AM NOW HOLDING
[run time = 2.0 sec]

33. when did you pick it up?
WHEN DID YOU DO PICKUP BLOCK(ANS:↑EXIST/X/ ← LAST ∞)?

I DID IT WHILE I PUT THE SMALL GREEN PYRAMID THAT
I AM NOW HOLDING ONTOP THE SMALL RED CUBE
[run time = 0. 4 sec]

32. when did you first pick it up?
WHEN DID YOU DO [ACT:↑GRASP/HAND(MAX)/BLOCK(ANS:↑EXIST/
X/ ← LAST ∞) → FIRST ∞]?

I DID IT WHILE I STACKUP THE BIG RED PARALLELEPIPED
THE BIG GREEN CUBE THAT IS NOW ON TOP OF A BIG RED
PARALLELEPIPED AND THE SMALL RED CUBE
[run time = 1.0 sec]

{Note that MAX answered in terms of the blocks that he actually stacked up rather than in terms of what he was requested to do. }

33. why?
WHY DID YOU DO [ACT:↑// → PRES 2]?

IN ORDER TO PUT THE SMALL GREEN PYRAMID THAT I AM
NOW HOLDING OFFOF THE SMALL RED CUBE
[run time = 0.1 sec]

{MAX remembers why he performed each action. }

34. why did you clear off that cube?
WHY DID YOU DO [ACT:↑OFFOF//BLOCK(KINDOF = CUBE)
→ PRES 4]?

IN ORDER TO PUT THE SMALL RED CUBE ONTOP THE BIG
GREEN CUBE THAT IS NOW ON TOP OF A BIG RED PARALLELEPIPED
[run time = 0.3 sec]

35. why did you do that?
WHY DID YOU DO [ACT:↑// → PRES 2]?

SO I COULD MOVE MY HAND FROM LOCATION -422 -608 -460
TO LOCATION -75 -450 -460
[run time = 0. 2 sec]

36. why?
WHY DID YOU DO [ACT:↑// → PRES 2]?

SO THE GRASP-STATUS OF MY HAND WOULD EQUAL GRASPING
BLOCK(B5)
[run time = 0.1 sec]

37. why did you do that?

WHY DID YOU DO [ACT:↑// → PRES 2]?

SO I COULD MOVE THE SMALL RED CUBE FROM LOCATION
-75 -450 -460 TO LOCATION -310 -550 -110
[run time = 0.1 sec]

33. why did you do that?
WHY DID YOU DO [ACT:↑// → PRES 2]?

IN ORDER TO PUT THE SMALL RED CUBE ONTOP THE BIG
GREEN CUBE THAT IS NOW ON TOP OF A BIG RED PARALLELEPIPED
[run time = 0.2 sec]

39. why did you do that?
WHY DID YOU DO [ACT:↑// → PRES 2]?

BECAUSE YOU ASK ME TO
[run time = 0.03 sec]

40. how did you do it'?
HOW DID YOU DO [COM:↑// ← PRES ω]?

| RELEASE THE BIG BLUE PARALLELEPIPED
| MOVE MY HAND FROM -545 145 -260 TO -545 -255 -360
| GRASP THE EIG GREEN CUBE
| MOVE THE BIG GREEN CUBE FROM -545 -255 -360 TO -310 -550 -160
| PUT 'I-HE BIG GREEN CUBE ONTOP THE BIG RED PARALLELEPIPED
| RELEASE THE BIG GREEN CUBE
| MOVE MY HAND FROM -310 -550 -160 TO -75 -450 -410
| GRASP THE SMALL GREEN PYRAMID
| MOVE THE SMALL GREEN PYRAMID FROM -75 -450 -410 TO -422 -608 -460
| TAKE -i-HE SMALL GREEN PYRAMID OFFOF THE SMALL RED CUBE
| RELEASE THE SMALL GREEN PYRAMID
| MOVE MY HAND FROM -422 -608 -460 TO -75 -450 -460
| GRASP THE SMALL RED CUBE
| MOVE THE SMALL RED CUBE FROM -75 -450 -460 TO -310 -550 -110
| PUT THE SMALL RED CUBE ONTOP THE BIG GREEN CUBE
[run time = 0.5 sec]

(MAX remembers in detail how he carried out a command. }

41 . how many objects did you touch while you were doing it?
// HOW MANY OBJECT() YOU DO GRASP WHILE YOU DO
[COM:↑// ↔ PRES 2]?

3 THE BIG GREEN CUBE THAT IS NOW ON TOP OF A BIG
RED PARALLELEPIPED AND THE SMALL RED CUBE AND
THE SMALL GREEN PYRAMID THAT I AM NOW HOLDING
[run time = 0.4 sec]

{This question is more difficult than it first appears to be. Note all the possible variations such as (42)}.

42. how many red objects did you clear off before you did it?
// HOW MANY OBJECT(COLOR = RED) YOU DO OFFOF BEFORE
YOU DO [COM:↑// ↔ PRES 2]?

1 THE BIG RED PARALLELEPIPED
[run time = 0.4 sfc]

43. what did the red cube support before you started to clean it off?
// FIND OBJECT(BOT_STATUS = ONTOP BLOCK(COLOR = RED|
KINDOF = CUBE)) BEFORE YOU DO OBJECT0 OFFOF BLOCK(COLOR
= RED|KINDOF = CUBE)?

IT IS THE SMALL GREEN PYRAMID
[run time = 2.5 sec]

{MAX enters his imaginary world and retraces his steps to the configuration shown in Fig. 9. Then he answers the questions by examining the state of the imaginary world. }

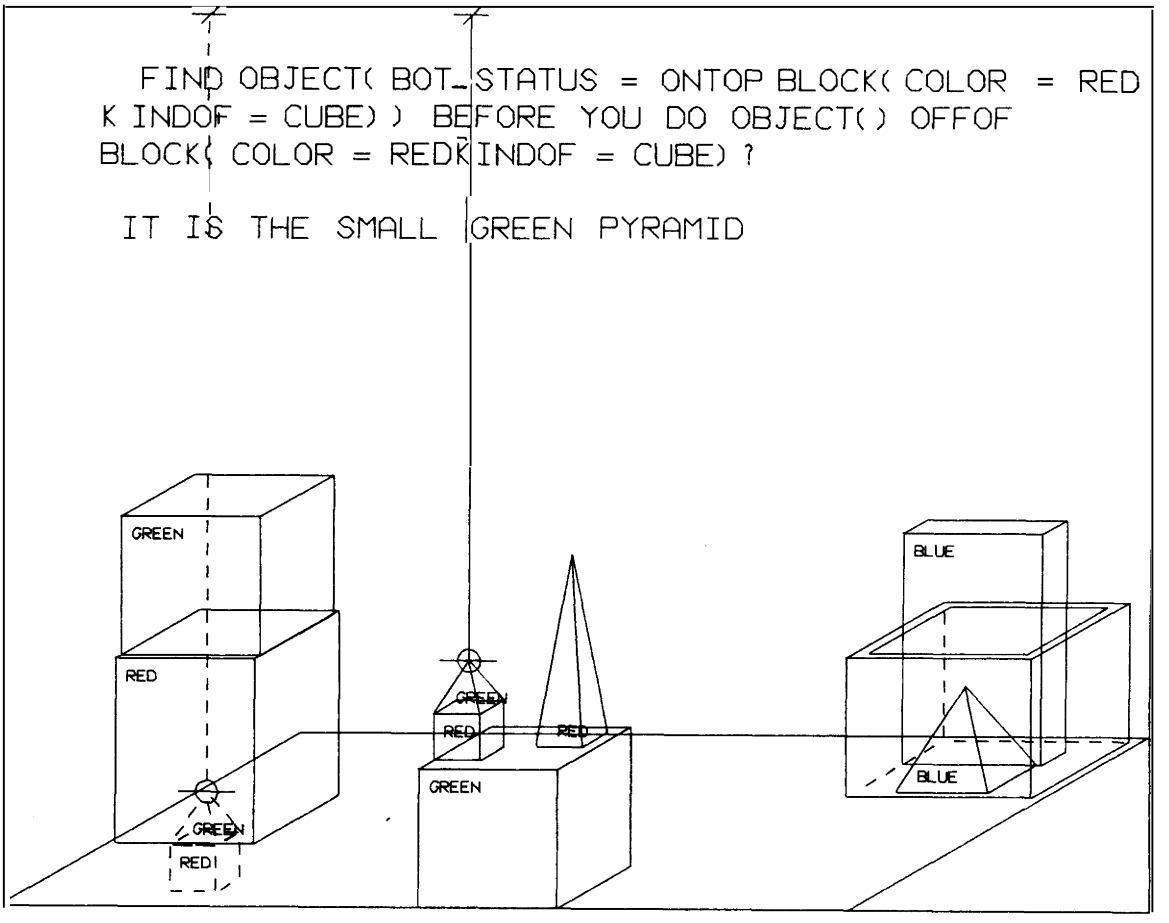
FIND OBJECT(BOT_STATUS = ONTOP BLOCK(COLOR = RED
K INDOF = CUBE)) BEFORE YOU DO OBJECT( ) OFFOF
BLOCK(COLOR = REDK INDOF = CUBE) ?
IT IS THE SMALL GREEN PYRAMID
GREEN
RED
GREEN
RED
GREEN
RED
RED
GREEN
BLUE
BLUE

Figure 9.

44. there were five blocks to the left of the box then.
// HOW MANY BLOCK(LOCASHUN LEFT-OF BOX(DEFINITE)|KINDOF = PARALLELEPIPED) AT THAT TIME?

4 THE BIG RED PARALLELEPIPED AND THE BIG GREEN CUBE THAT WAS ON TOP OF A BIG RED PARALLELEPIPED AND THE BIG GREEN CUBE THAT WAS HOLDING A SMALL RED PYRAMID AND THE SMALL RED CUBE
[run time = 3.3 sec]

45. how many things were on top of green cubes at that time?
// HOW MANY OBJECT(LOCASHUN DIRECTLY-ABOVE BLOCK(COLOR = GREEN|KINDOF = CUBE)) AT THAT TIME?

1 THE SMALL RED PYRAMID
[run time = 3.9 sec]

{MAX deleted the imaginary world after answering each of the questions (43), (44), and (45). This caused him to retrace his steps each time. If he had been a lit tle smarter, he could have saved the imaginary world from Question (43) to answer Questions (44) and (45). However, MAX was afraid that he would not know when to delete the imaginary world if he did that. }

46. put the blue pyramid on the block in the box.
// YOU GO BLOCK(COLOR = BLUE|KINDOF = PYRAMID) ONTOP BLOCK(BOT_STATUS = ONTOP BOX(DEFINITE)).

OK
[run time = 0.4 sec]

(The configuration of Fig. 10 is achieved. }

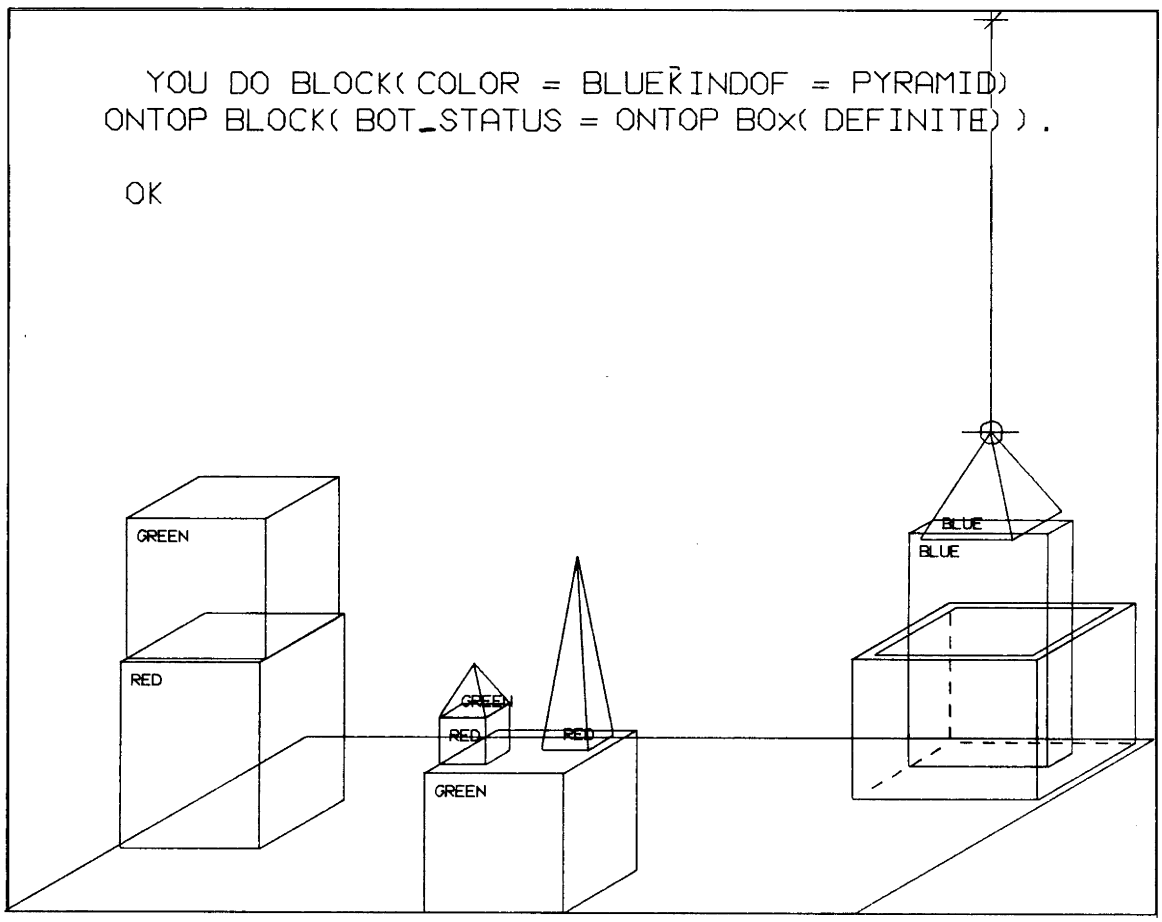
YOU DO BLOCK(COLOR = BLUE KINDOF = PYRAMID)
ONTOP BLOCK(BOT_STATUS = ONTOP BOX(DEFINITE)) .
OK
GREEN
RED
GREEN
RED
RED
GREEN
BLUE
BLUE

Figure 10.

47. is there anything which is bigger that every pyramid but is not as wide as the thing that supports it?
// DOES EXIST OBJECT(HEIGHT > HEIGHT[∀ BLOCK(KINDOF - PYRAMID)]|BOT_STATUS = ONTOP OBJECT(YWIDTH > YWIDTH[ITSELF]))?

YES THE BIG BLUE PARALLELEPIPED THAT I JUST PUT A SMALL BLUE PYRAMID ONTOP OF
[run time = 1. 7 sec]

{Note that MAX mentioned a recent event to help identify the object. }

48. does a steeple exist?
// DOES EXIST STEEPLEO?

NO BECAUSE COMPOSITE-OBJECT NOT EXIST STEEPLE
[run time = 0.02 sec]

49. a "steeple" is a stack which contains two green cubes and a pyramid.
// STORE COMPOSITE OBJECT STEEPLE0 = BLOCK(COLOR = GREEN| KINDOF = CUBE|TOP_STATUS = HOLDING BLOCK(COLOR = GREEN| KINDOF = CUBE|TOP_STATUS = HOLDING BLOCK(COLOR = RED| KINDOF = PYRAMID))).

OK
[run time = 0.02 sec]

[Our description of a "steeple" is essentially the description of the bottom block. }

50. are there any steeples now?
// DOES EXIST STEEPLEO?

NO BECAUSE BLOCK NOT EXIST BLOCK(COLOR = GREENIKINDOF = CUBE| TOP-STATUS = HOLDING BLOCK(COLOR = GREEN|KINDOF = CUBE|

TOP_STATUS = HOLDING BLOCK(COLOR = RED|KINDOF = PYRAMID)))
[run time = 0.3 sec]

(Note the different answers for the identical questions (48) and (50).)

51. build one.
YOU DO BUILD STEEPLE().

OK
[run time = 2.7 sec]

(MAX built the steeple shown in Fig. 11.)

52. how did you build the steeple?
// HOW DID YOU DO BUILD STEEPLE0?

I RELEASE THE SMALL BLUE PYRAMID
I MOVE MY HAND FROM -545 3.45 -160 TO -55 -155 -260
I GRASP THE SMALL GREEN PYRAMID
I MOVE THE SMALL GREEN PYRAMID FROM -55 -155 -260 TO -422 -608 -460
I TAKE THE SMALL GREEN PYRAMID OFFOF THE SMALL RED CUBE
I RELEASE THE SMALL GREEN PYRAMID
I MOVE MY HAND FROM -422 -608 -460 TO -55 -155 -320
I GRASP THE SMALL RED CUBE
I MOVE THE SMALL RED CUBE FROM -55 -155 -310 TO -472 -608 -460
I TAKE THE SMALL RED CUBE OFFOF THE BIG GREEN CUBE
I RELEASE THE SMALL RED CUBE
I MOVE MY HAND FROM -472 -608 -460 TO -105 -70 -160
I GRASP THE SMALL RED PYRAMID
I MOVE THE SMALL RED PYRAMID FROM -105 -70 -160 TO -522 -608 -310
I TAKE -I-HE SMALL RED PYRAMID OFFOF THE BIG GREEN CUBE
I RELEASE THE SMALL RED PYRAMID
I MOVE MY HAND FROM -522 -608 -310 TO -310 -550 -160
I GRASP THE BIG GREEN CUBE
I MOVE THE BIG GREEN CUBE FROM -310 -550 -160 TO -75 -105 -210
I PUT THE BIG GREEN CUBE ONTOP THE BIG GREEN CUBE

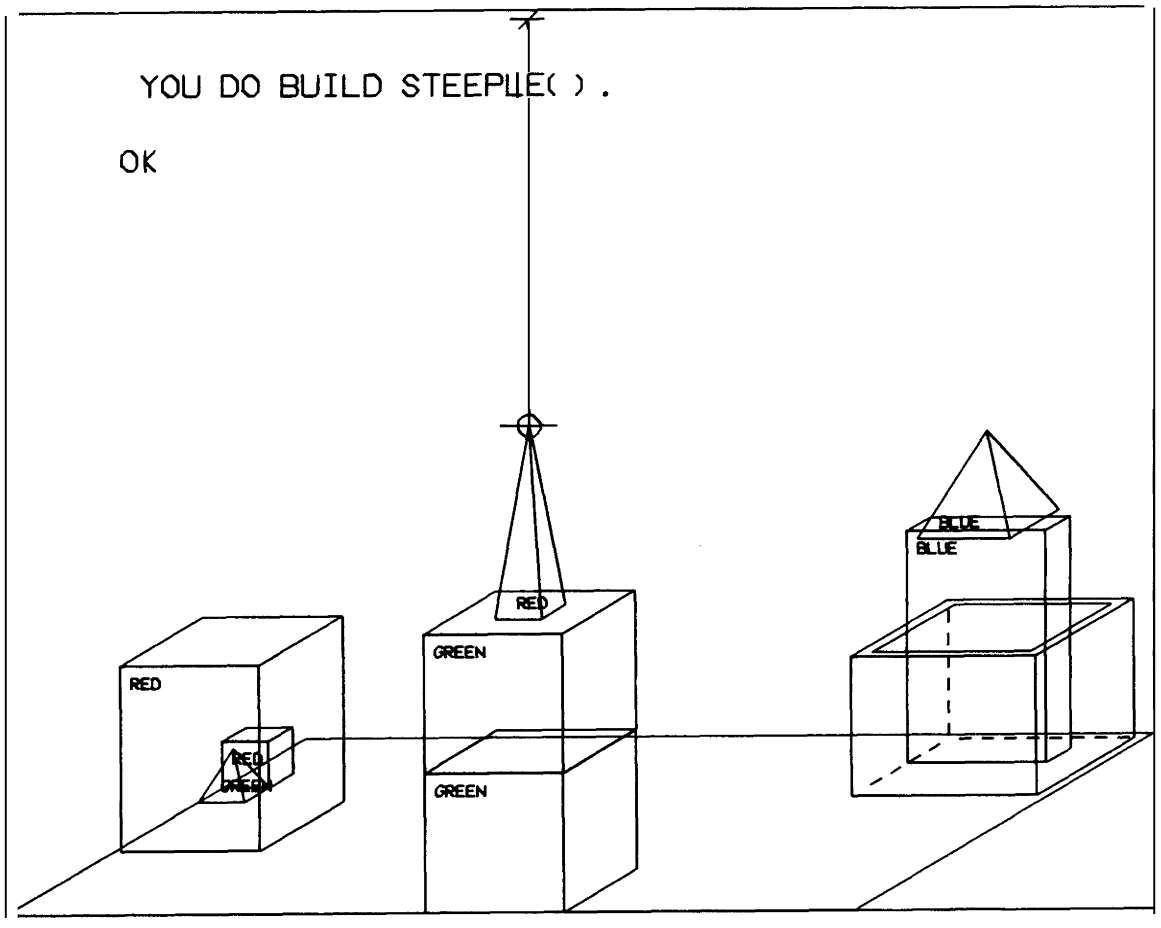

Figure 11.

I RELEASE THE BIG GREEN CUBE
I MOVE MY HAND FROM -75 -105 -210 TO -522 -608 -310
I GRASP THE SMALL RED PYRAMID
I MOVE THE SMALL RED PYRAMID FROM -522 -608 -310 TO -75 -105 -10
I PUT THE SMALL RED PYRAMID ONTOP THE BIG GREEN CUBE
[run time = 0. 9 sec]

53. a pile is a red pyramid which is on top of a blue block which ⊗
is supported by a green block.
// STORE COMPOSITE OBJECT PILE(GREEN|BLUE|RED) = BLOCK(COLOR
= GREEN|KINDOF = PARALLELEPIPEDITOP-STATUS = HOLDING
BLOCK(COLOR = BLUE|KINDOF = PARALLELEPIPEDITOP-STATUS
= HOLDING BLOCK(COLOR = RED|KINDOF = PYRAMID))).

OK
[run time = 0.03 sec]

{This definition for a composite object allows us to make substitutions. }

54. are they any green, green, and red piles now? ⊗
DOES EXIST PILE(GREEN|GREEN|RED)?

YES THE BIG GREEN CUBE THAT IS NOW HOLDING A BIG GREEN CUBE
[run time = 1.2 sec]

55. how would you build a red, green, and blue pile? ⊗
// HOW WOULD YOU DO BUILD PILE(RED|GREEN|BLUE)?

I WOULD MOVE THE SMALL RED PYRAMID FROM -75 -105 -10
TO -522 -608 -310
I WOULD TAKE THE SMALL RED PYRAMID OFFOF THE BIG GREEN CUBE
I WOULD RELEASE THE SMALL RED PYRAMID
I WOULD MOVE MY HAND FROM -522 -608 -310 TO -75 -105 -210
I WOULD GRASP THE GIG GREEN CUBE
I WOULD MOVE THE BIG GREEN CUBE FROM -75 -105 -210 TO -310 -550 -160
I WOULD PUT THE BIG GREEN CUBE ONTOP THE BIG RED PARALLELEPIPED

I WOULD RELEASE THE BIG GREEN CUBE
I WOULD MOVE MY HAND FROM -310 -550 -160 TO -545 145 -160
I WOULD GRASP THE SMALL BLUE PYRAMID
I WOULD MOVE THE SMALL BLUE PYRAMID FROM -545 145 -160
TO -310 -550 -60
I WOULD PUT THE SMALL BLUE PYRAMID ONTOP THE BIG GREEN CUBE
[run time - 3.4 sec]

{To answer this question, MAX entered his imaginary world, built the pile (see Fig. 12), and then examined how he did it in the imaginary world. }

56. get rid of those two green cubes.
// SUBTRACT DATABASE BLOCK(COLOR = GREEN|KINDOF = CUBE|
EOT,STATUS = ONTOP BLOCK(COLOR = GREEN)) AND SUBTRACT
DATABASE BLOCK(COLOR = GREENIKINDOF = CUBE).

OK
OK
[run time = 0.5 sec]

{Note that MAX cleared off the top green cube before he destroyed it. See Fig. 13.}

57. add a black cylinder and a very tall orange cylinder.
// ADD DATABASE BLOCK(COLOR = BLACK|KINDOF = CYLINDER)
AND ADD DATABASE BLOCK(COLOR = ORANGE|KINDOF = CYLINDER1
HEIGHT ≥ HEIGHT[∀ BLOCK()]).

OK
OK
[run time = 0.4 sec]

{The two cylinders appear as in Fig. 14. }

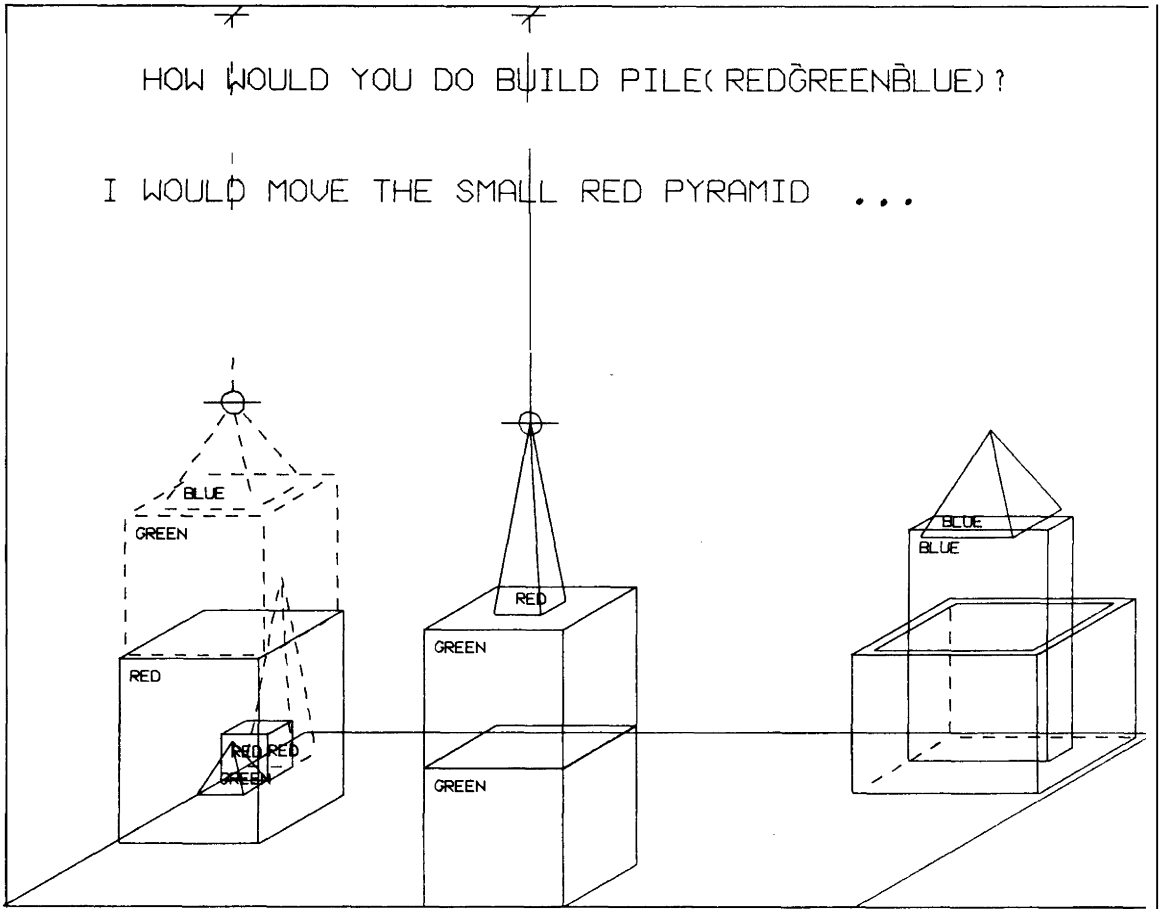
HOW WOULD YOU DO BUILD PILE(REDGREENBLUE)?
I WOULD MOVE THE SMALL RED PYRAMID ...
BLUE
GREEN
RED
RED RED
GREEN
RED
GREEN
GREEN
BLUE
BLUE

Figure 12.

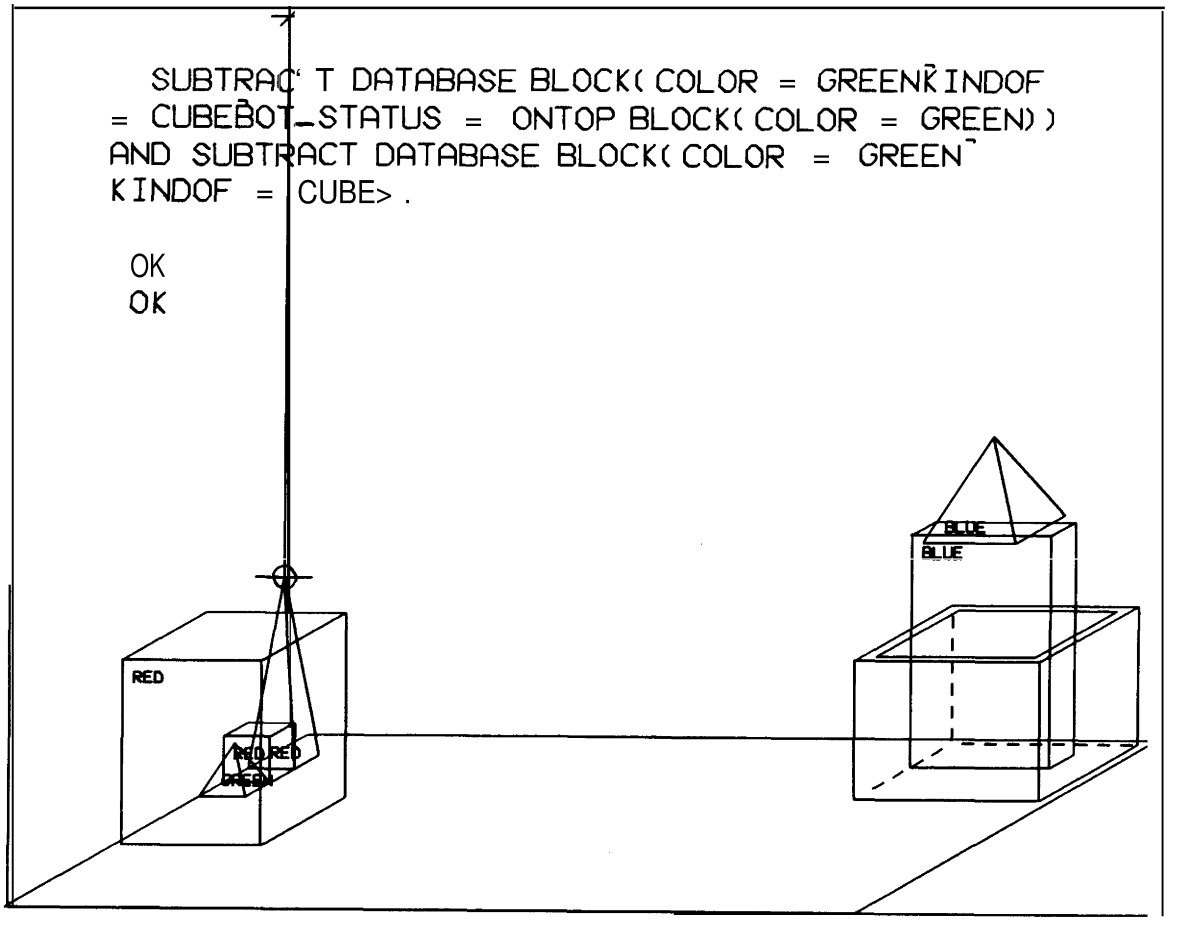
SUBTRACT DATABASE BLOCK(COLOR = GREEN KINDOF
= CUBE BOT_STATUS = ONTOP BLOCK(COLOR = GREEN))
AND SUBTRACT DATABASE BLOCK(COLOR = GREEN
KINDOF = CUBE> .
OK
OK
RED
BLUE

Figure 13.

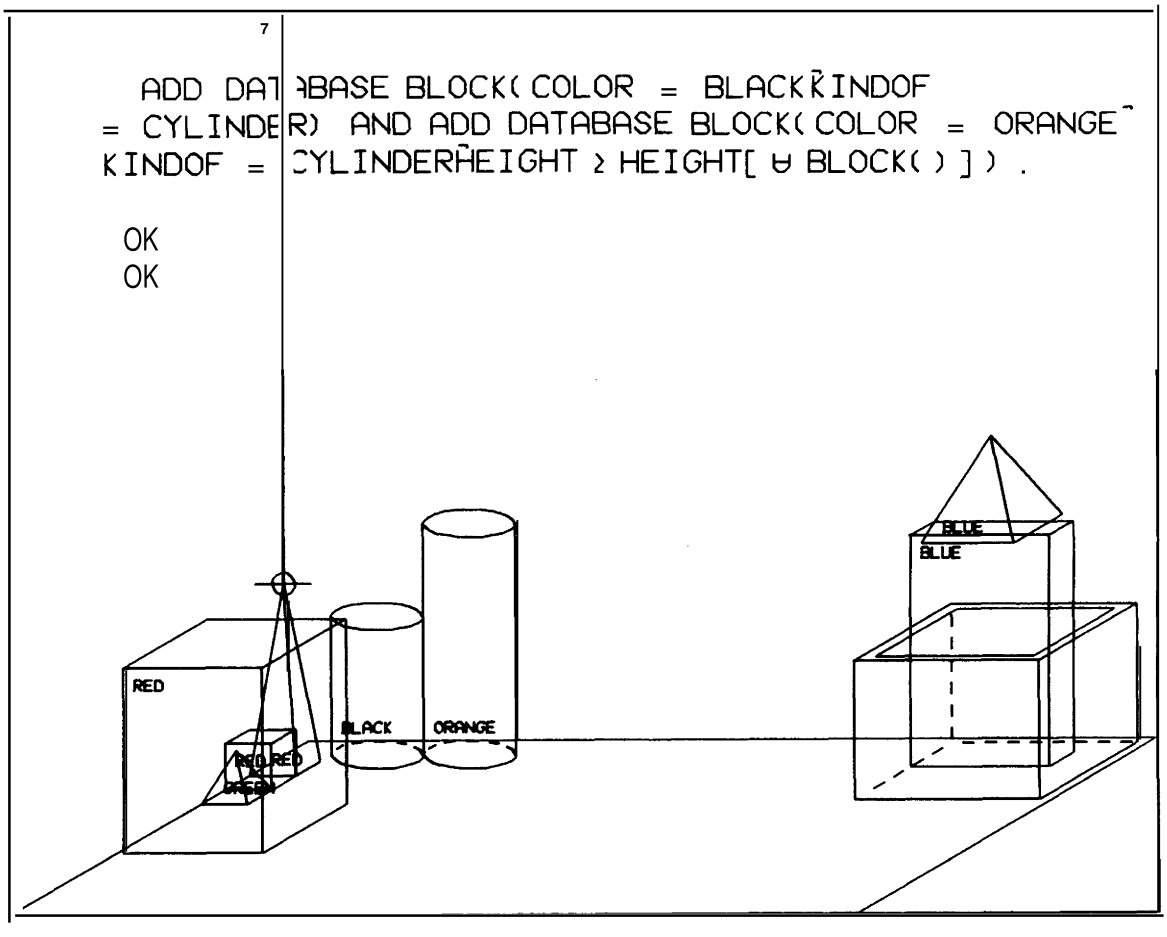
7
ADD DAT ABASE BLOCK(COLOR = BLACK KINDOF
= CYLINDER) AND ADD DATABASE BLOCK(COLOR = ORANGE
KINDOF = CYLINDER HEIGHT > HEIGHT[U BLOCK( )]) .
OK
OK
RED
BLACK
ORANGE
BLUE
BLUE

Figure 14.

58. add a small yellow cone in your hand and a box on top of the large red block.

```
# ADD DATABASE BLOCK(COLOR = YELLOW|KINDOF = CONE|SIZE
= SMALL|HELD_STATUS = GRASPED-BY HAND(MAX)) AND ADD
DATABASE BOX(DESCRIPTION = THE BOX|BOT_STATUS = ONTOP
BLOCK(COLOR = RED|SIZE = BIG)).

OK
OK
[run time = 0.3 sec]
```

{MAX released -the red pyramid so he would have a free hand for the cone. Note that MAX wilt follow any requirements given in the description but he also fills in needed data that is not specified. The result is shown in Fig. 15. }

59. a rocket made of parallelepipeds and a pyramid is a pile.

```
// STORE COMPOSITE OBJECT ROCKET(PARALLELEPIPED|PYRAMID|
YELLOW|BROWN|GREEN) = PILE(YELLOW|BROWN|GREEN).

OK
| run time = 0.7 sec]
```

{Note how easy it is to tell MAX about a rocket since he already knows what a pile is. }

60. build a rocket of orange and black cylinders and a yellow cone.

```
I: YOU DO BUILD ROCKET(CYLINDER|CONE|ORANGE|BLACK|YELLOW)?

OK
[run time = 2.8 sec]
```

{The result is shown in Fig. 16. }

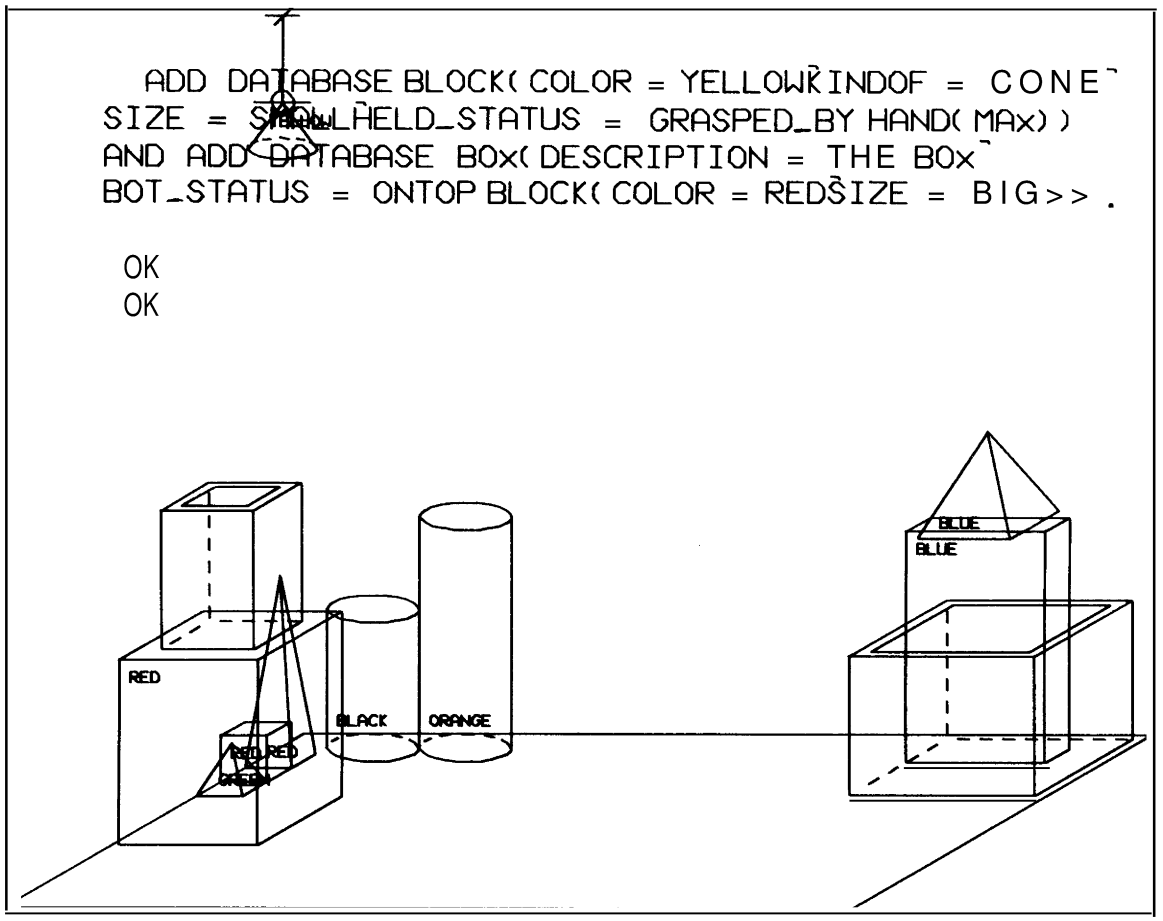
ADD DATABASE BLOCK(COLOR = YELLOW KINDOF = CONE
SIZE = SMALL HELD_STATUS = GRASPED_BY HAND(MAx))
AND ADD DATABASE BOx(DESCRIPTION = THE BOx
BOT_STATUS = ONTOP BLOCK(COLOR = RED SIZE = BIG>> .
OK
OK
RED
BLACK
ORANGE
BLUE
BLUE

Figure 15.

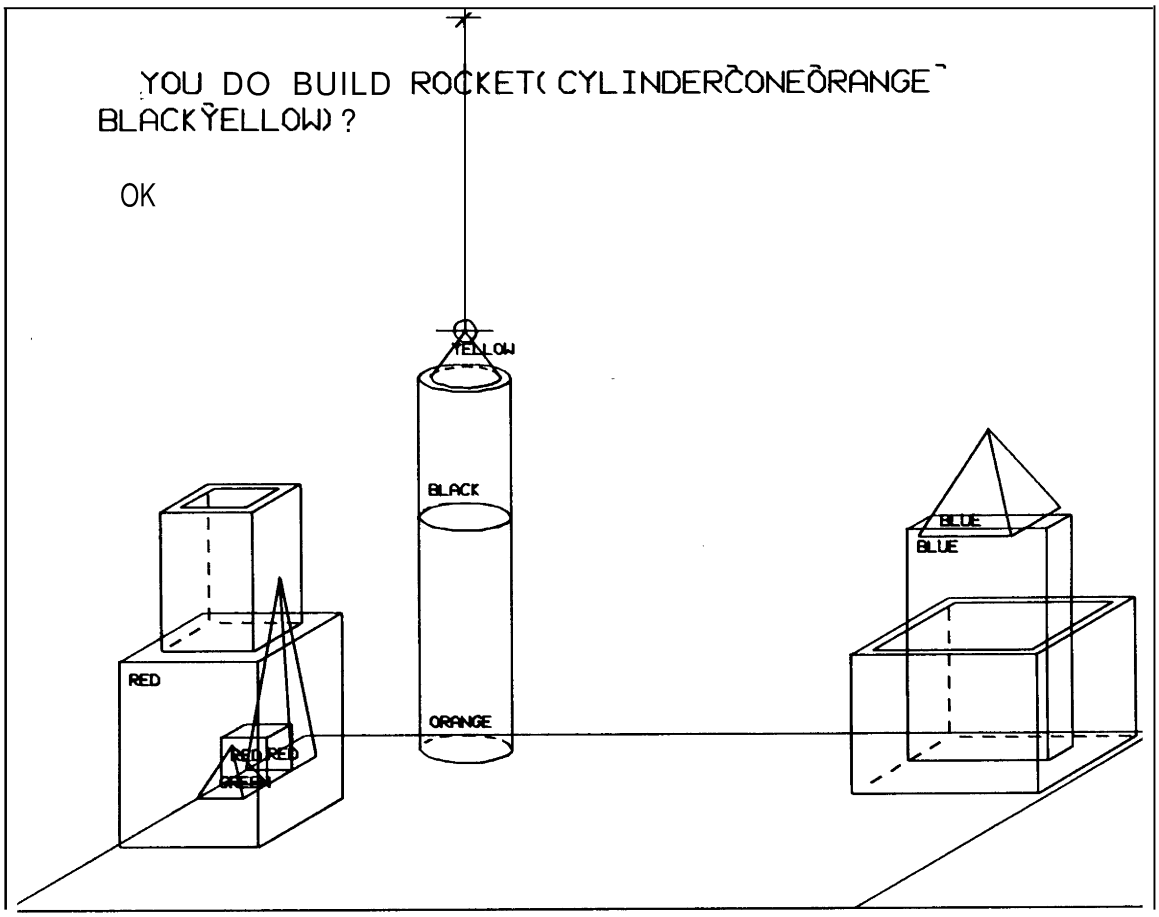
YOU DO BUILD ROCKET(CYLINDER^CONE^ORANGE^
BLACK^YELLOW)?
OK
YELLOW
BLACK
ORANGE
RED
BLUE
BLUE

Figure 16.

61. what objects are on top of the table now?
FIND OBJECT(BOT_STATUS = ONTOP TABLE(DEFINITE))?

IT IS THE BIG RED PARALLELEPIPED AND THE SMALL
GREEN PYRAMID AND THE SMALL RED CUBE AND THE SMALL
RED PYRAMID AND THE MEDIUM-SIZED ORANGE CYLINDER
THE BOX THAT IS NOW HOLDING A BIG BLUE PARALLELEPIPED
[run time = 1.2 sec]

[Note that MAX must now identify which box he is refering to. }

62. call the biggest block "superblock".
STORE FACT GNAME OF BLOCK(VOLUME ≥ VOLUME[V BLOCK()])
IS SUPERBLOCK.

OK
[run time = 0.2 sec]

{We can establish between ourselves and MAX, a common name for any object. }

63. have you picked up superblock since we began?
WHEN DID YOU DO PICKUP BLOCK(GNAME = SUPERBLOCK)?

I DID IT WHILE I PICKUP THE BIG RED PARALLELEPIPED
CALLED SUPERBLOCK
[run time = 1.5 sec]

{MAX still has his old internal name for that object, so he is not confused in searching his past. }

64. how many blocks other than pyramids were left of the box at that time?
HOW MANY BLOCK(KINDOF ≠ PYRAMID|LOCASHUN LEFT-OF
BOX()) AT THAT TIME?

5 THE BIG BLUE PARALLELEPIPED AND THE BIG RED
PARALLELEPIPED CALLED SUPERBLOCK AND THE SMALL RED
CURE AND THE BIG GREEN CUBE THAT WAS HOLDING A SMALL
RED PYRAMID AND THE BIG GREEN CUBE WHICH WAS TOWARD
THE BACK
[run time = 14.8 sec]

{This question is a difficult challenge for any representation system. MAX must not only retrace his steps, but he must also destroy and resurrect old objects in an imaginary world as he does it. The result is shown in Fig. 17. }

65. why did you drop superblock?
// WHY DID YOU DO DROP BLOCK(GNAME = SUPERBLOCK)?

I DON'T REMEMBER DOING THAT
[run time = 3.2 sec]

66. is there anything in front of the red pyramid?
// DOES EXIST OBJECT(LOCASHUN IN-FRONT-OF BLOCK(COLOR = RED|
KINDOF = PYRAMID))?

YES THE BIG RED PARALLELEPIPED CALLED SUPERBLOCK
AND THE SMALL GREEN PYRAMID AND THE SMALL RED CUBE
THE BOX THAT IS NOW ON TOP OF A BIG RED PARALLELEPIPED
[run time = 1.0 sec]

(See Fig. 18. }

67. thank you.
// THANK YOU.

YOU' RE WELCOME

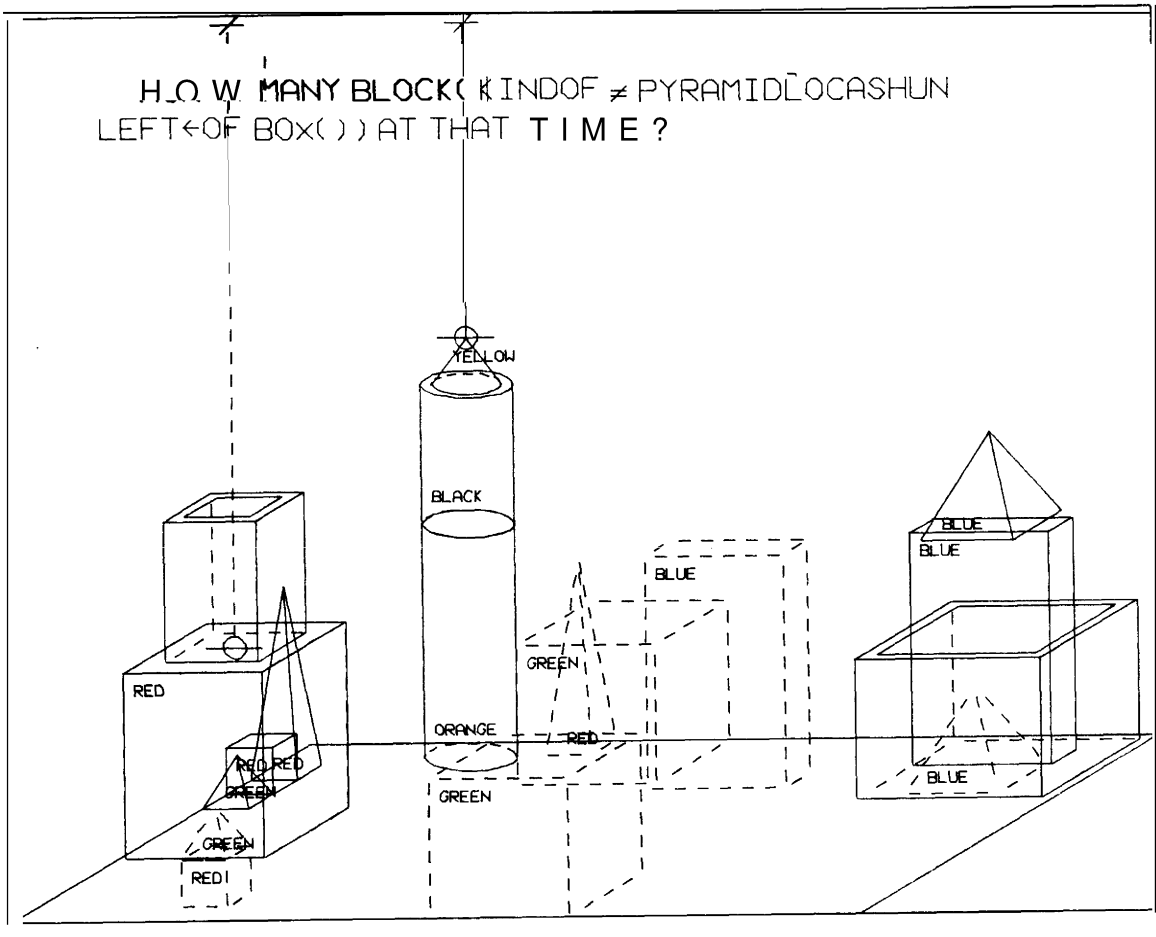
HOW MANY BLOCK(KINDOF ≠ PYRAMIDLOCASHUN
LEFT←OF BOX( )) AT THAT TIME ?
YELLOW
BLACK
BLUE
BLUE
BLUE
GREEN
RED
ORANGE
RED
RED RED
GREEN
GREEN
GREEN
BLUE
RED

Figure 17.

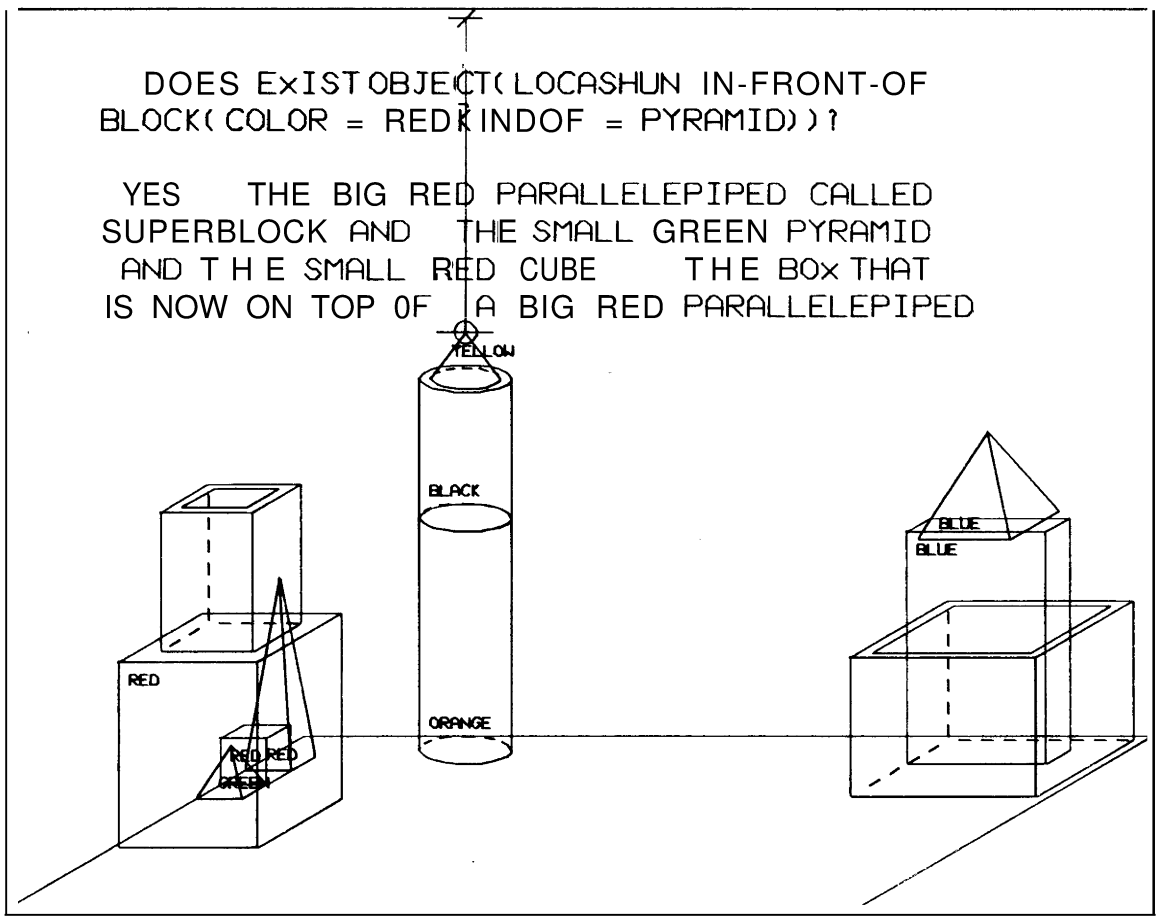
DOES EXIST OBJECT(LOCASHUN IN-FRONT-OF
BLOCK(COLOR = RED) KINDOF = PYRAMID))?
YES THE BIG RED PARALLELEPIPED CALLED
SUPERBLOCK AND THE SMALL GREEN PYRAMID
AND THE SMALL RED CUBE THE BOX THAT
IS NOW ON TOP OF A BIG RED PARALLELEPIPED
YELLOW
BLACK
BLUE
BLUE
RED
ORANGE

Figure 18.

III. DATABASES

It is convenient to split the database into two parts, one which contains complete information about the world in its present state (Knowledge Represent at ion Database) and another which contains historic information about the past conversations and events (Historic Database). Data for the imaginary world is stored in the same string arrays (and same locations) as that for the real world.

A. Knowledge Representation Database

It is convenient to group objects in one's world into categories. In this robot's world there are only five categories:

1 BLOCK
2 BOX
3 TABLE
4 HUMAN
5 HAND

Here HAND stands for the Robot whose only effector is a hand. This database is a list of properties. For each category there are two types of lists. One is a list of properties which pertains to all objects in that category, while the other is a list of properties which pertains to tokens or particular objects. Thus there is one double-index array for each category. The first index goes from 0 to some number (for example 10). The property list pertaining to ail objects of the category is stored in the array with the first index = 0, while the other numbers are used for storing the property lists of tokens. A few examples should make all of this clear.

Table I. Property List for "All" BLOCK's

BLOCK[0,PNAME]	IS	BLOCK
BLOCK[0,STATUS]	IS	8
BLOCK[0,FIRST_ONE]	IS	1
BLOCK[0,MOVEABLE]	IS	MOVABLE
BLOCK[0,AN_VEG_MIN]	IS	VEGETABLE
BLOCK[0,ITSELF_LIST]	IS	1
BLOCK[0,PLIST]	IS	2
BLOCK[0,MAX_NUMB_POSSIBLE]	IS	15
BLOCK[0,KINDOF]	IS	WOODEN
BLOCK[0,GNAME]	IS	

Table II. Property List for a Particular BLOCK

BLOCK[4,PNAME]	IS	B4
BLOCK[4,HELD_STATUS]	IS	FREE
BLOCK[4,LOCASHUN]	IS	-75 -105 -360
BLOCK[4,COLOR]	IS	GREEN
BLOCK[4,SIZE]	IS	BIG
BLOCK[4,DESCRIPTION]	IS	THE BIG GREEN CUBE
BLOCK[4,TOP_STATUS]	IS	HOLDING BLOCK(B6)
BLOCK[4,BOT_STATUS]	IS	ONTOP TABLE(TABL1)
BLOCK[4,KINDOF]	IS	CUBE
BLOCK[4,DIMENSIONS]	IS	150 150 150
BLOCK[4,XLENGTH]	IS	150
BLOCK[4,YWIDTH]	IS	150
BLOCK[4,HEIGHT]	IS	150
BLOCK[4,XCOORD]	IS	-75
BLOCK[4,YCOORD]	IS	-105
BLOCK[4,ZCOORD]	IS	-360
BLOCK[4,WALL_WIDTH]	IS	
BLOCK[4,LIKED_STATUS]	IS	
BLOCK[4,GNAME]	IS	
BLOCK[4,VOLUME]	IS	3375000
BLOCK[4,DISP_NUMB]	IS	4
BLOCK[4,SHAPE_OF_TOP]	IS	FLAT

Table III. Property List for a Particular HUMAN

HUMAN[1,PNAME]	IS	FRIEND
HUMAN[1,GRASP_STATUS]	IS	EMPTY
HUMAN[1,WEIGHT]	IS	175
HUMAN[1,LOCASHUN]	IS	100 200 200
HUMAN[1,HAIR_COLOR]	IS	BROWN
HUMAN[1,SIZE]	IS	BIG
HUMAN[1,DESCRIPTION]	IS	YOU
HUMAN[1,TOP,STATUS]	IS	CLEAR
HUMAN[1,BOT_STATUS]	IS	ONTOP CHAIR(CHAIR1)
HUMAN[1,KINDOF]	IS	HACKER
HUMAN[1,EYE,COLOR]	IS	BROWN
HUMAN[1,AGE]	IS	25
HUMAN[1,SEX]	IS	MALE
HUMAN[1,HEIGHT]	IS	72
HUMAN[1,XCOORD]	IS	100
HUMAN[1,YCOORD]	IS	200
HUMAN[1,ZCOORD]	IS	200
HUMAN[1,LIKE_STATUS]	IS	
HUMAN[1,LIKED_STATUS]	IS	
HUMAN[1,GNAME]	IS	
HUMAN[1,VOLUME]	IS	
HUMAN[1,DISP_NUMB]	IS	
HUMAN[1,SHAPE_OF_TOP]	S	ROUND

The property list for all blocks is shown in Table I, and the property list for a particular token, in Table II, and the property list of a particular human, in Table III. The two indices for the string arrays are shown in square brackets. The first index is a nurnber, while the second index is a string (MACRO) which is translated by the compiler into a number. The array contains the strings in the right-hand column. You might ask "Where does the semantic information reside? It is true that you have property lists, but what do the properties mean to the program?" The answer is that the semantics resides in the procedures which know about these properties. The string names of the properties are contained in the string array NAMEOF[I,J] for which the first index refers to which list (different categories have different lists--the number for these are stored under <object>[0,PLIST]--as shown in Table I) and the second index corresponds to the property number.

Certain properties are classified as "unchangeable" or "related". A property may be in one, both, or neither of these classifications. An unchangeable property cannot be changed by telling (i. e. the command: STORE I-ACT . . .). Examples are XCOORD and TOP-STATUS. A related property is one that has a complement. Examples are: (1) TOP-STATUS, BOT,STATUS; (2) LIKE-STATUS, LIKED-STATUS. If a change in a related property occurs, this results in other changes to the database. For example, if the BOT,STATUS of some object is changed, this results in the TOP-STATUS of some other object being changed. (There is a double index array which holds related properties and their complements and a short procedure which will quickly obtain the complement of any property.) New properties can readily be added from the teletype (see Section VIIC and VIIE).

B. Historic Database

Information about commands, facts, questions, answers, actions, reasons, inferences, orders (internal commands), adding new tokens, subtracting tokens, and thoughts about all of the above are stored in a single index array called the "Grapevine". Many procedures store information on this "Grapevine" Array while some examine this information, In order to establish a time sequence, an array with only one index is used. This simple method works well in general but some problems such as simultaneous events cannot be handled so simply.

Table IV. Grapevine Array Format

CQM: FACT: QUEST: ANS: ACT: REAS: INFER: ORD: ADD: sue: THOUG H-f :COM: THOUGHT:FACT:	→ {extra information) ↑<action>/<subject>/<object>

The general format for an entry is shown in Table IV. The string before the colon identifies the type of information. The string between the colon and the upward arrow contains extra information such as type of question if it is a question, who said it to whom, or the name of a procedure to identify itself. Between the arrow and first slash there is an action word such as a verb (or "not" followed by a verb). The next two strings are <subject> and <object>. Most of the entries fit this format, but there are a few exceptions such as the action "move" which has "from" and "to" information. Typical entries can be seen in APPENDIX A which contains t hc Grapevine Array which the program generated during the dialog of Section II.

IV. PROCEDURAL DESCRIPTIONS

Objects are identified by a category narne followed by a description in parentheses. Typical examples of descriptions are given in Table V. As one can see there are several different formats. This flexibility seems to be necessary. BLOCK(B3) is an example in which the format is just the PNAME ii. e. the program's name for that particular object). This type of description is used by the program as a fast, definite description. In principle a human communicating with MAX would never know or refer to this name. If a human wants to establish between himself and MAX a common name for an object, he uses the property GNAME, which can be changed without upsetting the databases.

BLOCK(VAR B3 B4 B7) is an example of a description which refers to any or all of the blocks with PNAMES B3, B4, and B7. This is useful in cases for which there exists a choice. If MAX were asked to pick up a green block and B3, B4, and B7 were green blocks, the program would convert the description to this form and delay a definite decision until the last moment.

The most common type of description has the format of:

<property> <relation> <state>.

Table V. Examples of descriptions

BLOCK(B3)

BLOCK(COLOR = RED|KINDOF = CUBE)

BLOCK(VAR B3 B4 B7)

BLOCK(COLOR = BLUE|HEIGHT >
HEIGHT[BLOCK(HELD_STATUS = GRASPED-BY HAND(MAX))])

OBJECT(COLOR = GREEN|BOT_STATUS = ONTOP
OBJECT(HEIGHT ≤ HEIGHT[∀ OBJECT(BOT_STATUS = ONTOP
OBJECT(TOP_STATUS = HOLDING BLOCK(KINDOF = PYRAMID|
HEIGHT ≥ HEIGHT[∀ BLOCK(KINDOF = PYRAMID)])))]))

BLOCK(KINDOF ≠ PYRAMID|ANS:↑EXIST/X/ ← 107 ω)

BLOCK(KINDOF = PYRAMID|ACT:↑GRASP/HAND(MAX)/X ←
COM:↑ONTOP/BLOCK(COLOR = GREEN|KINDOF = PYRAMID)
/BLOCK(SIZE = SMALL|KINDOF = CUBE>+ ∞)

BLOCK(VAR B3 B4 B7) is an example of a description which refers to any or all of the blocks with PNAMES B3, B4, and B7. This is useful in cases for which there exists a choice. If MAX were asked to pick up a green block and B3, B4, and B7 were green blocks, the prograrn would convert the description to this form and delay a definite decision until the last moment.

The most common type of description has the format of:

<property> <relation> <state>.

Table VI. Examples of Descriptive Segment

Property	Relation	State
	(a)	
KINDOF	=	PYRAMID
TOP_STATUS	=	HOLDING BLOCK(. . .)
HEIGHT	≥	HEIGHT[BOX(. . .)]
	(b)	
YCOORD		YCOORD[∀ BLOCK(. . .)]
HEIGHT		HEIGHT[∃ BLOCK(. . .)]

Examples are shown in Table VI. If the program were asked to find BLOCK(KINDOF = PYRAMID), it would cycle through all of its block tokens pulling out their property KINDOF and noting those for which this property were equal to PYRAMID. Some properties are more complicated such as TOP-STATUS. For example, the TOP_STATUS of a particular block might be

HOLDING BLOCK(B4)|HOLDING BOX(BOX3)|HOLDING BLOCK(B2)

while the state that the prograrn is trying to match is

HOLDING BLOCK(COLOR = BLUE).

The comparison is done by first finding all blocks satisfying the descript ion COLOR = BLUE through a recursive call by the procedure on it self. Then each block that it is holding is cornpared with all the blue

blocks. The program does an exhaustive search returning all objects satisfying any description, rather than stopping after it finds the first one.

To contrast our descriptions with those of Winograd, consider the description "a red cube which supports a pyramid". Winograd's[20] description (PLANNER program) is:

```
(GOAL (IS ?X1 BLOCK))
(GOAL (COLOR-OF ?X1 RED))
(GOAL (EQUIDIMENSIONAL ?X1 ))
(GOAL (IS ?X2 PYRAMID))
(GOAL (SUPPORT ?X1 ?X2))
```

while our description is:

```
BLOCK(COLOR = RED|KINDOF = CUBE|
TOP-STATUS = HOLDING BLOCK(KINDOF = PYRAMID)).
```

The <state> in third example in Table VI(a) has the form:

<property>[<object>(<description>)].

The program (1) finds the particular object (if there is more than one and the state is not quantified, it reports failure); (2) gets the value of <property> for this object and converts it to a number. Then this is compared with the <property> (also converted to a number) according to the designated relation.

The program uses the following relations:

=, ≠, ≥, ≤, <, >, ABOVE, BELOW, LEFT-OF, RIGHT-OF, BEHIND, IN-FRONT-OF, DIRECTLY-ABOVE, DIRECTLY-BELOW, EQUALS-EITHER-OR, and NOT <relation>.

There is a short algorithm connected with each of these that gives them an exact mathematical (although not necessarily intuitive> meaning.

Quantified descriptions have the form shown in Table VI(b). The

familiar predicate calculus symbols "∀" and "∃" all used for “all” and "some". However, their effect on the calculation is only that caused by a particular algorithm and any symbol could be used. The combined effect of the relation and the quantifier determines the number used for <state> in the comparison. For example if the relation is "≤" and the quantifier is "∀" then we must find the smallest number.

Any number of descriptions (<property> <relation> <state>) can be concatenated together with a vertical line for a delimiter as shown in the fifth example of Table V. Also, as indicated in that example the description can contain other descriptions to any depth.

Another format is used to describe objects which were previously referred to. Pattern matching of the historic Grapevine Array is done by filling in some (or possibly none) of the locations and by putting an "X" in the location of the desired quantity as in the description:

BLOCK(ANS:↑ EXIST/X/ ← LAST ∞).

I-his type of description has the format:

<type>:<before-uparrow>↑<action>/<subject>/<object>
<arrow> <pointer> <how-far>.

The first part is just the format for an entry in the Grapevine Array. The quantity <pointer> indicates where the search should start and it can be: (1) FIRST, for starting at the first entry; (2) LAST, for starting at the last entry; (3) PRES, for starting at the time marker for the present discussion. The words “when”, “how”, and “why” cause a time marker to be set to some Grapevine index number, and PRES refers to this number; (4) some Grapevine index number; or (5) another Grapevine entry to be matched as in the last example of Table VI.

The quantity <arrow> can be ←, →, or ↔ depending upon whether the search is to be backward, forward, or around the designated entry. The quantity <how-fat+ is a nurnber indicating the extent of the search (∞ means go to the end).

A null (or "X") is the description will match anything in the

corresponding field of the Grapevine entry. Non-null fields in <type>, <before-uparrow>, and <action> must be identical for a match. In matching <subject> and <object> fields we note that

COM:HUMAN(FRIEND) TO HAND(MAX)↑PICKUP/HAND(MAX)/
BLOCK(COLOR = RED|SIZE = BIG)

should be matched by

COM:↑PICKUP/X/BLOCK(COLOR = RED).

Also

ACT:↑GRASP/HAND(MAX)/BLOCK(B7)

should be matched by

ACT:X↑GRASP//BLOCK(KINDOF = PYRAMID)

if the block with PNAME of B7 is a pyramid.

These matches are accomplished as follows: (1) The set of all objects satisfying the desired description is found; (2) The set of all objects satisfying the Grapevine description is found; (3) A set intersection of the two sets is performed; (4) If the resulting set is not empty, they match. Otherwise, they do not match.

Matches are also done to find locations in the Grapevine Array. In these cases the "X" is omitted.

In matching the total description, the general order is that each object is tested against the description until it fails to satisfy some requirement or it succeeds. Descriptions involving actions such as "red objects that you touched while . . . " (see Questions 41 and 42) are handled by: (1) finding all objects satisfying the static description; (2) finding all objects satisfying the motion description; and (3) doing a set intersection of the two results. A description of the form:

BLOCK(KINDOF = PYRAMID|ACT:↑GRASP/HAND(MAX)/X ← LAST ∞)

is handled by backtracking (programmed especially for this case>. It would be inefficient to search the Grapevine for all objects that the hand has grasped and yet the program would fail (without backtracking) if the first object that it found was not a pyramid.

The description format for composite objects was chosen so that it would be easy to identify the presence of a composite object. For example,

PILE(GREEN|BLUE|RED) = BLOCK(COLOR = GREEN|
KINDOF = PARALLELEPIPEDITOP,STATUS = HOLDING BLOCK(COLOR = BLUE|
KINDOF = PARALLELEPIPED|TOP_STATUS = HOLDING BLOCK(COLOR = RED|
KINDOF = PYRAMID)))

is really only the description of one object (the bottom object) for which the program already has an identification mechanism. The description inside the parenthesis following the name of the composite object is treated like a set of variables in a macro definition. Any quantity put inside the parenthesis on the left (at the time that the definition is given) can be freely substituted for in its every occurrence on the right at a later time.

V. MOTION PROCEDURES

There are a set of procedures for moving objects:

Specialists

GRASP
RELEASE
GETONTOPOF
GETOFFOF
FIND-SPOT
MOVETO

MOVE_DIRECT
AVOID_COLLISION

Strategists

CHANGETO
STACKUP
BUILD

One can command MAX to do the following simple actions (involving one or two objects and the hand): grasp, release, ontop, offof, putdown, pickup, near, inside, and move. All such requests pass through the procedure CHANGETO which simply calls upon one or two of the specialists. The specialists are rather independent. They check the present state of the world at the time that they are called and report failure if any errors in syntax or inconsistencies appear. If the present state of the world is ready for them to do their job, they simple do it and exit. However, if the present state of the world does not permit them to do their job, they call on other procedures and themselves recursively to create the proper conditions. For example, if we have the conditions shown in Fig. 10, and one were to command MAX to grasp the green cube which is sitting on the table, the procedure GRASP would call on (1) GETOFFOF so that the object it wants to grasp would have a clear top; (2) RELEASE so its hand would be empty; and (3) MOVETO so its hand would be at the correct location to grasp the green cube. This simple operation (GRASP is the only procedure called by CHANGETO) would cause the following calling sequence:

```
21  GRASP       {the big green cube)
12  GETOFFOF     {the small red cube off of the big green cube)
6   GETOFFOF     {the small green pyramid off of the small red cube]
3   GRASP       {the small green pyramid)
1   RELEASE     {the small blue pyramid}
2   MOVETO       (hand frorn the blue pyramid to the small green pyramid]
4   FIND-SPOT    {for the small green pyramidj
5   MOVETO       {the small green pyramid to the table]
9   GRASP       {the small red cube)
7   RELEASE      (the small green pyramid)
```

```
8  MOVETO      (hand from the green pyramid to the small red cube)
10 FIND-SPOT   {for the small red cube)
11 MOVETO      {the small red cube to the table)
18 GETOFFOF    (the small red pyramid off of the big green cube)
15 GRASP       (the small pyramid)
33 RELEASE     {the small red cube)
14 MOVETO      {hand from the small red cube to the small red pyramid)
16 FIND-SPOT   {for the small red pyramid}
17 MOVETO      (the small red pyramid to the table)
19 RELEASE     {the small red pyramid}
20 MOVETO      {hand from the small red pyramid to the big green cube}
```

Note that the order in which procedures are entered (listed from top to bottom) is different from the order in which they do their main job (given by the numbers on the left). The main effect of procedure GRASP (and RELEASE) is to change the GRASP-STATUS of the hand and the HELD_STATUS of some object, but their side effects can be considerable as we have just discussed. RELEASE will not do its job unless the object to be released is supported. If it is not, RELEASE will check to see if there is really some object just below the one that it wants to release. If there is, it calls upon another procedure to rnodify the top and bottom status of the objects involved. If not, it calls on GETONTOPOF to put the object on the table.

The location of objects is changed only by procedure MOVETO or MOVE-DIRECT. (MOVE-DIRECT and AVOID_COLLISION are used in special cases in which the hand and anything it happens to be holding are moved left, right, up, down, backward, or forward). The main effect of GETONTOPOF and GETOFFOF are to change the top and bottom status of objects. All locations for placing objects ontop of other objects or the table are selected by the procedure FIND-SPOT, although other procedures can suggest that it use a certain location.

The procedures STACKUP and BUILD examine the objects, plan a strategy for the overall task (without considering details), and then they make calls on CHANGETO.

This method of moving objects is not new; Winograd[6,20] used a

very similar met hod. The advantage of this method is that the higher-level procedures need only worry about the task that they want to achieve and not the grimy details.

VI. CONTEXT MECHANISM

In order to answer hypothetical questions, the program needs to carry out actions in an imaginary world. These actions must not affect the data for the real world as the program will eventually want to return quickly to this state. Usually one wants to start the modifications with the database in its present form in the real world. Making an extra copy of the database is an unsatisfactory solution as this would require a long time and a large memory space for a system with a large database. Typically one wants the real-world database (which could be very large) with only a small number of modifications.

This same problem also arises when the program wants to know about a past state of the world. Saving all past states of the world is out of the question. However, even if one knows what modifications occurred and the order that they occurred in, he still needs an extra database (which is initially identical to the real database) in which to make the changes.

To solve these and other similar problems, we have devised a context mechanism. It is not too different from those used in CONNIVER[13], QA4[12], AND MLISP2[21] when one considers the great difference in programming languages.

In implementing this context mechanism, we require that all transfers of information to and from the Knowledge Representation Database pass through a filter. The filtering procedure checks the present context and takes the appropriate action. If the context+0 then some locations in string arrays may contain data which is different in the real world from that in the imaginary world. The two data. strings are separated by a "*" with the imaginary-world data on the left and the real-world data on the right. Data with no "*" are identical in the two worlds. A typical BLOCK array for such a case is shown in Table VII.

Table VII. Property List for a Particular BLOCK with Existence of Imaginary Context

BLOCK[5,PNAME]	IS	B5
BLOCK[5,HELD_STATUS]	IS	FREE*FREE
BLOCK[5,LOCASHUN]	IS	-310 -550 -260*-75 -450 -460
BLOCK[5,COLOR]	IS	RED
BLOCK[5,SIZE]	IS	SMALL
BLOCK[5,DESCRIPTION]	IS	THE SMALL RED CUBE
BLOCK[5,TOP_STATUS]	IS	HOLDING BLOCK(B6)*HOLDING BLOCK(B7)
BLOCK[5,BOT_STATUS]	IS	ONTOP BLOCK(B2)*ONTOP TABLE(TABL1)
BLOCK[5,KINDOF]	IS	CUBE
BLOCK[5,DIMENSIONS]	IS	50 50 50
BLOCK[5,XLENGTH]	IS	50
BLOCK[5,YWIDTH]	IS	50
BLOCK[5,HEIGHT]	IS	50
BLOCK[5,XCOORD]	IS	-310*-75
BLOCK[5,YCOORD]	IS	-550*-450
BLOCK[5,ZCOORD]	IS	-260*-460
BLOCK[5,WALL_WIDTH]	IS	
BLOCK[5,LIKED_STATUS]	IS	
BLOCK[5,GNAME]	IS	
BLOCK[5,VOLUME]	IS	125000
BLOCK[5,DISP_NUMB]	IS	7*5
BLOCK[5,SHAPE_OF_TOP]	IS	FLAT

The rules used in filtering the data to and from the string arrays are as follows:

(1) if the context=0, then the filter does nothing letting data flow in the normal manner.

(2) if the context=1, then:

(a) data is taken from the right of the "*" if a "*" exists. Otherwise, from the complete location as normal.

(b) data is stored after the "*" if a "*" exists. If no "*" exists, then one is added before storage and a note of this location is made.

(3) if the context=-1, then:
(a) data is taken from the left of the "*" if a "*" exists. Otherwise, from the complete location as normal.
(b) data is stored before the "*" if a "*" exists. If no "*" exists, then one is added before storage and a note of this location is made.

The addresses of string-array locations which contain imaginary-world data that is different than real-world data (i.e. a "*" is present) are saved. (Note that each address in only saved once.) Thus the effort involved in returning to the state in which only real-world data exists (context = 0) is proportional to the extent of the modifications in the imaginary world and involves only a change in the "*" locations.

This context mechanism could be generalized to several contexts by using several delimiters or adding cont ext labels between the entries, but its efficiency would suffer greatly. At present, we have not found the need for many contexts. The context mechanism has been implemented to handle the case in which one request, "Suppose we only had one pyramid and one cube, what . . . ". Here we do not want to waste time adding a "*" to all locations in setting up the imaginary world. This is handled by setting the context = -2 for which only entries with a "*" exist in the imaginary world. This context frame is convenient for small, completely different, imaginary worlds.

Note that the frame with context = +1 or -1 can be used to handle the situation in which one requests, "Remember everything as it is now. All right, make the following changes . . . ". Here one just changes the context from 0 to +1 and the state which the robot was asked to remember will be the imaginary world with context = -1.

In summary, we think that this is a useful context mechanism for a robot because (1) there is essentially no overhead involved in changing to an imaginary world, and (2) The additional storage space and time involved are proportional to the size of the changes in the imaginary world.

VII. DISCUSSION

In this section we shall discuss several short topics which did not seem to be appropriate for any earlier section.

A. Self-debugging

As Winograd[22] and others have noted, with large programs such as SHRDLU there is a complexity barrier making them difficult to understand and extend. When one wants to add a new procedure or modify an old one, he may not remember all the conditions and requirements of other sections of the program (particularly if several months have elapsed since the other sections were written).

With this complexity problem in mind, MAX was written in what some might call an inefficient manner with considerable redundancy. This slightty increased the programming time and the size of the program, but it greatly reduced the debugging time. All procedures (except trivial ones) were given some independence. Each procedure has some expectancy about its input data. If the syntax is wrong or the data is in any way inconsistent with these expectations, the procedure reports an error and indicates the form of the error by adding an entry to the Grapevine Array. If the procedure should fail to achieve an objective, it must also report failure with a Grapevine message telling why the failure occurred. This error testing greatly aided debugging because the error was detected earlier and the program was less likely to die. For example if some new input "tickled a bug" the error would usually be detected either inside the procedure in which it occurred or the next procedure.. Whereas, if no error testing were done, the program might ramble on in its recursive, interwoven manner through a dozen procedures before the error was detected or it died. The message usually explained the cause of the problem. If not, the program was still alive to answer more questions about the bug.

B. Generating Answers

It is much easier to generate natural language than to parse natural language into the correct program representation. Although the program does not put out particularly good English, the effort for generating this out put was trivial. This leads one to believe that it is a rather straightforward problem to generate output that humans can understand. The same cannot be said of handling natural-language input which is a far more difficult problern. (To generate output that is indistinguishable from that of hurnans is, of course, difficult[23].) Some answers are "canned", but most answers are a concatenation of strings from various parts of the program. For example, the answers to Questions 12, 13, and 14 are "canned". If the program gets to one of these places in its analysis, there is only one concept that it wishes to convey so a "canned" answer seems appropriate.

Procedure IDENTIFY provides a complete description for an object (its input is the object's category and pname). First, it obtains a short description such as "the big green cube" from the property DESCRIPTION (see Table II). Then it checks to see if the object has a gname. If it does, then "called <gname>" is added to the short description. If it has no gname, then the property HELD-STATUS is checked. If it is being held in robot' s hand then "that I STRING1 holding" is added to the description for which STRING1 is "am now" or "was" depending upon the situation. If neither of the above conditions apply then IDENTIFY calls upon a procedure which interrogat es the Grapevine Array to find out if this object has been involved in any recent actions. This procedure interrogates the Grapevine Array for six entries back and reports what action, if any, this object was involved in. This is responsible for such answers as:

IT IS THE BOX THAT I JUST PUT A BIG BLUE PARALLELEPIPED INSIDE OF (Quest ion 5)

YES THE BIG BLUE PARALLELEPIPED THAT I JUST PUT A SMALL BLUE PYRAMID ONTOP OF (Question 47).

If none of the above apply, the program checks to see if any other object has the same description. If none does, it is satisfied with the short description. If the object' s description is identical to that of one or more other objects, the procedure IDENTIFY checks the properties TOP-STATUS and BOT_STATUS trying to find something to distinguish it from other similar objects. If these fail, it determines the direction with the greatest spatial variation to separate this object from similar objects. This leads to answers such as,

5 THE . . . THE BIG GREEN CUBE THAT WAS HOLDING A SMALL RED PYRAMID AND THE BIG GREEN CUBE WHICH WAS TOWARD THE BACK (Question 64)

For all failures the program calls on procedure REASON which puts together a string by examining the Grapevine. For example consider Question 2. The Grapevine contains the following reason for failure:

GRAPEVINE 22: REAS: FIND_NUMB/TOOMANY↑FIT/DESCRIPTION/
BLOCK(VAR B6 B7 B8)

Procedure FIND-NUMB has found three blocks satisfying the description while the DEFINITE in the description required that it only find one. Therefore it reported failure and gave the above reason. Procedure REASON just put this into the normal form of subject-verb-object, calling on procedure IDENTIFY to describe the blocks with pnames B6, B7, and B8. Thus reason returns the string:

BECAUSE DESCRIPTION FIT THE SMALL RED PYRAMID
THE SMALL GREEN PYRAMID AND THE SMALL BLUE PYRAMID

C. Reasoning

As one can see from the dialog (Section II), the program has considerable reasoning power, but no explicit predicate calculus is used. Instead the program' s reasoning abilities are built into the procedures and

structure of the system. (In other words, we followed the PLANNER[11] method.) For example consider the following simple reasoning problem in which one gives the program two factual statements and then asks a quest ion:

All humans are mortal.
Turing is a human.
Is Turing mortal?

The following dialog was generated with MAX (since MAX does not have the property MORTAL-STATUS stored, we must first get it added to the property list for humans):

STORE PROPERTY.

OK -- WHAT CATEGORY OF OBJECTS IS IT A PROPERTY OF? (BLOCK,BOX,HUMAN,...?)

HUMAN

IS IT A PROPERTY OF ALL HUMAN'S -- Y OR N?

YES

WHAT IS THE NAME OF THE PROPERTY?

MORTAL-STATUS

IS IT AN UNCHANGEABLE PROPERTY? Y OR N?

NO

IS IT RELATED PROPERTY Y OR N?

// NO

I' M READY TO STORE THE FOLLOWING DATA:
PROPERTY LIST = 3
PROPERTY = MORTAL-STATUS

ARE ALL OF THESE CORRECT -- Y OR N?

// YES

OK

// STORE FACT MORTAL-STATUS OF HUMAN(ITSELF) IS MORTAL.

OK

// ADD DATABASE HUMAN(GNAME = TURING).

OK

#ISFACT MORTAL-STATUS OF HUMAN(GNAME = TURING) IS MORTAL?

YES

// IS FACT MORTAL-STATUS OF HUMAN(GNAME = TURING) IS IMMORTAL?

NO BECAUSE MORTAL-STATUS NOT EQUAL IMMORTAL

// WHAT IS MORTAL-STATUS OF HUMAN(GNAME = TURING)?

MORTAL

Note that the program actually added a "token" human to its database, and its ability to answer the question was due to its use of the database structure.

D. Creating, Destroying, and Resurrecting Tokens (Particular Objects)

Unlike Winograd's SHRDLU[6], MAX can readily create and destroy tokens (see Questions 56-58). The program has a set of default values for properties so one can just command:

ADD DATABASE BLOCK0

and the program will choose the type of block, its size, and call upon FIND-SPOT to find a location for it, etc. Note that the program can use its motion procedures to clear off objects before it destroys them and put down objects so it has a free hand for new objects.

Whenever a token is destroyed, all of the information about the token is eliminated from the Knowledge Representation Database (its position in the string array is given to another token). However, the token's complete property list is saved (in the usual descriptive format) in a Grapevine entry. Thus it is easy to resurrect that token at a later time. To answer Question 64, the program in an imaginary context destroyed. objects and resurrected two green cubes. Since their pnames were unique, the program had no trouble moving a resurrected token in retracing its steps.

It is quite apparent that a human taking MAX's part in the dialog would not (although he could) make a complete representation of the scene at the beginning and then carry out the operation blind by modifying his

representation. It is more likely that he would look at the scene between actions, filling his database with tokens (from visual input) each time. Thus the addition and deletion of tokens from the working database may be very important in simulating human behavior--particular if vision is included.

One sometimes wonders how a large Knowledge Representation Database could work efficiently. For example, if one asked MAX, "Who is the tallest person in the room?" and MAX knew 1000 people, he would cycle through the all the people (1000) in his Knowledge Representation Database before answering which would be very inefficient. One possible solution to this problem is to have a small, relevant, working Knowledge Representation Database. In this case, for example, it might only be filled to answer the particular question and therefore only contain those people in the room. The properties of all other tokens could be contained in something like the Grapevine Array. Thus, when two old friends appear, they (their property lists, that is) could be resurrected to the working database.

E. Suggestions for Future Work

Since it took only six months (see Section I) to get the program to this level, one can be optimistic about extending it. As long as the program runs rapidly and the self-debugging is effective, the larger, the better--that is, it is easier to do some new process because one can call on so many old procedures. Also it is easier for the program to learn about some new composite object if they can be described in terms of ones that it already knows about. For example, MAX was told about a "rocket" in terms of a "pile".

With the program at its present level of competence, there are several interesting directions in which it could be extended:

(1) Increasing the representation to handle properties and relations that are needed in a more complex world such as Euler Angles to specify the orientation of an object and concepts such as objects "touching";

(2) Adding some or all of the vision segments shown in Fig. 1;

(3) Adding a content-addressable database with pointers to entries in the Grapevine Array;

(4) Increasing the abilities of the executive to do inferencing[24,25] and examination of all error messages;

(5) Extending the Grapevine Array format to something like a conceptual-dependency diagram[7,24,25];

(6) Increasing the program' s learning ability. At present, this involves storing facts, adding and deleting tokens and properties. The adding of new properties is perhaps the highest level of learning achieved in the program, and it gives some indication of the methods necessary for ext ending this capability. Properties can be added directly from the teletype (see Section VIIC) and the program asks the necessary questions. A complicated property such as LIKE-STATUS (and its complement) are handled by putting strings in several arrays and increasing the count number in other arrays. This learning ability should be extented to adding and deleting new categories, new relations, and new actions.

Having procedural knowledge can make it more difficult to add new knowledge. (Sussman[26] has worked on this problem.) For example, if one adds a new type of object, the program needs to know the property SHAPE-OF-TOP. If this property is "round", "pointed", or "flat" then the present program can handle it. However, if it were "peaked", one would have to modify the coding to handle this new case. This and similar problems could be handled by storing lists (long strings that contain entries separated by a break character and which can be augmented) of acceptable and unacceptable SHAPE-OF-TOP s for supporting other objects.

VIII. ACKNOWLEDGMENTS

We would like to thank T. Winograd, J. A. Feldman, and T.O. Binford for helpful discussions, J. R. Low for helping with SAIL related problems, and B. A. Perkins for the graphic display program.

APPENDIX A. HISTORIC DATABASE GENERATED BY DIALOG

The Historic Database is initially empty when a conversation begins. This section contains the Grapevine-Array entries which the program generated during the dialog of Section II. So that one can easily find the Grapevine entries for a particular question, the computer output has been edited by the insertion of question numbers. "INFO" refers to array INFO whose entries are given at the end. The program usually refers to objects by their pnames which for the blocks are as follows:

B1 IS THE BIG BLUE PARALLELEPIPED.
B2 IS THE BIG RED PARALLELEPIPED.
B3 IS THE BIG GREEN CUBE WHICH INITIALLY IS ON TOP OF BLOCK B2.
B4 IS THE BIG GREEN CUBE WHICH INITIALLY IS HOLDING BLOCK B6.
B5 IS THE SMALL RED CUBE.
B6 IS THE SMALL RED PYRAMID.
B7 IS THE SMALL GREEN PYRAMID.
B8 IS THE SMALL BLUE PYRAMID.

---------- Question 1 ----------

GRAPEVINE 1.: COM:HUMAN(FRIEND) TO HAND(MAX)↑PICKUP/HAND(MAX)/BLOCK(COLOR = RED|SIZE = BIG)
GRAPEVINE 2: ACT:↑MOVE/HAND(MAX)/HAND(MAX)/FROM 0 0 0/TO -31.0 -550 -160
GRAPEVINE 3: REAS:ACHIEVE↑GRASP/HAND(MAX)/BLOCK(B3)
GRAPEVINE 4: ACT:CAUSED_BY HAND(MAX)↑GRASP/HAND(MAX)/BLOCK(B3)
GRAPEVINE 5: REAS:ACHIEVE↑OFFOF/BLOCK(B3)/BLOCK(B2)
GRAPEVINE 6: ORD:GETOFFOF TO FIND_SPOT↑FIND/FIND_SPOT/INFO//1
GRAPEVINE 7: ACT:↑MOVE/HAND(MAX)/BLOCK(B3)/FROM -310 -550 -160/TO -545 -255 -360
GRAPEVINE 8: REAS:ACHIEVE↑ONTOP/BLOCK(B3)/TABLE(TABL1)
GRAPEVINE 9: ACT:CAUSED_BY HAND(MAX)↑OFFOF/BLOCK(B3)/BLOCK(B2)
GRAPEVINE 10: REAS:ACHIEVE↑TOP_STATUS[BLOCK(B2)] = CLEAR

GRAPEVINE 11: ACT:CAUSED,BY HAND(MAX)↑RELEASE/HAND(MAX)/
BLOCK(B3)
GRAPEVINE 12: REAS:ACHIEVE↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -545 -255 -360/TO -310 -550 -310
GRAPEVINE 13: ACT:↑MOVE/HAND(MAX)/HAND(MAX)/FROM -545
-255 -360/TO -310: -550 -310
GRAPEVINE 14: REAS:ACHIEVE↑GRASP_STATUS[HAND(MAX)]
= GRASPING BLOCK(B2)
GRAPEVINE 15: ACT:CAUSED,BY HAND(MAX)↑GRASP/HAND(MAX)/
BLOCK(B2)
GRAPEVINE 16: REAS:ACHIEVE↑MOVE/HAND(MAX)/BLOCK(B2)/
FROM -310 -550 -310/TO -310 -550 200
GRAPEVINE 17: ACT:↑MOVE/HAND(MAX)/BLOCK(B2)/FROM -310
-550 -310/TO -310 -550 200
GRAPEVINE 18: REAS:↑ASK/HUMAN(FRIEND)/HAND(MAX)
GRAPEVINE 19: ANS:HAND(MAX) TO HUMAN(FRIEND)↑DID/HAND(MAX)/
INFO#2

---------- Question 2 ----------

GRAPEVINE 20: COM:HUMAN(FRIEND) TO HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(DEFINITE|KINDOF = PYRAMID)
GRAPEVINE 21: ACT:NOAP
GRAPEVINE 22: REAS:FIND_NUMB/TOOMANY↑FIT/DESCRIPTION/
BLOCK(VAR B6 B7 B8)
GRAPEVINE 23: ANS:HAND(MAX) TO HUMAN(FRIEND)↑NOT DID/
HAND(MAX)/INFO#3

---------- Question 3 ----------

GRAPEVINE 24: COM:HUMAN(FRIEND) TO HAND(MAX)↑FIND/
HAND(MAX)/BLOCK(HEIGHT > HEIGHT[BLOCK(HELD_STATUS =
GRASPED-BY HAND(MAX))])
GRAPEVINE 25: ANS:HAND(MAX) TO HUMAN(FRIEND)↑FIND/
HAND(MAX)/BLOCK(B1)
GRAPEVINE 26: COM:HUMAN(FRIEND) TO HAND(MAX)↑INSIDE/
BLOCK(ANS:↑ FIND//X → 23 2)/BOX(DEFINITE)
GRAPEVINE 27: ORD:GETONTOPOF TO FIND_SPOT↑FIND/FIND_SPOT/

INFO#4
GRAPEVINE 28: ORD:GETONTOPOF TO FIND_SPOT↑FIND/FIND_SPOT/
INFO#5
GRAPEVINE 29: ACT:↑MOVE/HAND(MAX)/BLOCK(B2)/FROM -310
-550 200/TO -310 -550 -310
GRAFEVINE 30: REAS:ACHIEVE↑ONTOP/BLOCK(B2)/TABLE(TABL1)
GRAPEVINE 31: ACT:CAUSED,BY HAND(MAX)↑ONTOP/BLOCK(B2)/
TABLE(TABL1)
GRAPEVINE 32: REAS:ACHIEVE↑RELEASE/HAND(MAX)/BLOCK(B2)
GRAPEVINE 33: ACT:CAUSED_BY HAND(MAX)↑RELEASE/HAND(MAX)/
BLOCK(B2)
GRAFEVINE 34: REAS:ACHIEVE↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -310 -550 200/TO -475 -105 -260
GRAPEVINE 35: ACT:↑MOVE/HAND(MAX)/HAND(MAX)/FROM -310
-550 200/TO -475 -105 -260
GRAPEVINE 36: REAS:ACHIEVE↑GRASP_STATUS[HAND(MAX)]
= GRASPING BLOCK(B1)
GRAPEVINE 37: ACT:CAUSED_BY HAND(MAX)↑GRASP/HAND(MAX)/
BLOCK(63)
GRAPEVINE 38: REAS:ACHIEVE↑MOVE/HAND(MAX)/BLOCK(B1)/
FROM -475 -1.05 -260/TO -545 145 -260
GRAPEVINE 39: ACT:↑MOVE/HAND(MAX)/BLOCK(B1)/FROM -475
-105 -260/TO -545 145 -260
GRAPEVINE 40: REAS:ACHIEVE↑ONTOP/BLOCK(B1)/BOX(BOX1)
GRAPEVINE 41: ACT:CAUSED_BY HAND(MAX)↑INSIDE/BLOCK(B1)/
BOX(BOX1)
GRAPEVINE 42: REAS:↑ASK/HUMAN(FRIEND)/HAND(MAX)
GRAPEVINE 43: ANS:HAND(MAX) TO HUMAN(FRIEND)↑DID/HAND(MAX)/
INFO#6

---------- Question 4 ----------

GRAPEVINE 44: COM:HUM AN(FRIEND) TO HAND(MAX)↑ FIND/
HAND(MAX)/OBJECT(BOT_STATUS = ONTOP BOX(DEFINITE))
GRAPEVINE 45: ANS:HAND(MAX) TO HUMAN(FRIEND)↑FIND/
HAND(MAX)/BLOCK(VAR B1 B8)

------- ____ -- Question 5 ----------

GRAPEVINE 46: COM:HUMAN(FRIEND) TO HAND(MAX)↑FIND/
HAND(MAX)/OBJECT(TOP_STATUS = HOLDING BLOCK(KINDOF = PYRAMID|
ANS:↑FIND//X ← 45 2))
GRAPEVINE 47: ANS:HAND(MAX) TO HUMAN(FRIEND)↑FIND/
HAND(MAX)/BOX(BOX1)

---------- Question 6 ----------

GRAPEVINE 48: QUEST:HOW_MANY/HUMAN(FRIEND) TO HAND(MAX)
'T EXIST/BLOCK(BOT,STATUS ≠ ONTOP BOX(DEFINITE)|KINDOF
= PARALLELEPIPED)/
GRAPEVINE 49: ANS:HAND(MAX) TO HUMAN(FRIEND)↑EXIST/
BLOCK(VAR B2 B3 B4 B5)/4

---------- Question 7 ----------

GRAPEVINE 50: QUEST:YES_NO/HUMAN(FRIEND) TO HAND(MAX)↑EXIST/
BLOCK(ANS:↑EXIST/X/ ← 49 2|YWIDTH < YWIDTH[BLOCK(COM:
HUMAN(FRIEND) TO HAND(MAX)↑PICKUP/HAND(MAX)/X ← 49 ∞)])/
GRAPEVINE 51: ANS:HAND(MAX) TO HUMAN(FRIEND)↑EXIST/
BLOCK(B5)/1

---------- Question 8 ----------

GRAPEVINE 52: QUEST:YES_NO/HUMAN(FRIEND) TO HAND(MAX)↑EXIST/
OBJECT(TOP_STATUS = HOLDING BLOCK(ANS:↑EXIST/X/ ← 51 2))/
GRAPEVINE 53: ANS:HAND(MAX) TO HUMAN(FRIEND)↑ EXIST/
TABLE(TABL1)/1

---------- Question 9 ----------

GRAPEVINE 54: QUEST:IS_POSSIBLE/HUMAN(FRIEND) TO HAND(MAX)
↑ DO/TABLE/PICKUP BLOCK()

GRAPEVINE 55: ANS:HAND(MAX) TO HUMAN(FRIEND)↑NO//
GRAPEVINE 56: REAS:↑NOT EQUAL/AN_VEG_MIN[TABLE(ITSELF)]/
ANIMATE

---------- Question 10 ----------

GRAPEVINE 57: QUEST:IS_POSSIBLE/HUMAN(FRIEND) TO HAND(MAX)
↑ONTOP/BLOCK(KINDOF = PYRAMID)/BLOCK(KINDOF = PARALLELEPIPED)
GRAPEVINE 58: ANS:HAND(MAX) TO HUMAN(FRIEND)↑YES//
GRAPEVINE 59: REAS:↑EXIST/EXAMPLE/INFO#7

---------- Quest ion 11 ----------

GRAPEVINE 60: THOUGHT:COM:HUMAN(FRIEND) TO HAND(MAX)↑ONTOP/
BLOCK(KINDOF = PYRAMID)/BLOCK(KINDOF = PYRAMID)
GRAPEVINE 61: THOUGHT:OHD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#8
GRAPEVINE 62: THOUGHT:ACT:NOAP
GRAPEVINE 63: THOUGHT:REAS:FIND_SPOT/CANNOT_BE_DONE↑NOT
SUPPORT/BLOCK(SHAPE-OF-TOP = POINTED)/OBJECT(VAR)
GRAPEVINE 64: ANS:HAND(MAX) TO HUMAN(FRIEND)↑NO//

---------- Question 12 ----------

GRAPEVINE 65: QUEST:IS_POSSIBLE/HUMAN(FRIEND) TO HAND(MAX)
↑DO/HAND(MAX)/STACKUP BLOCK(COLOR = RED) BLOCK(COLOR
= RED) AND BLOCK(COLOR = RED)
GRAPEVINE 66: THOUGHT:COM:HUMAN(FRIEND) TO HAND(MAX)
↑STACKUP/HAND(MAX)/BLOCK(COLOR = RED) BLOCK(COLOR =
RED) AND BLOCK(COLOR = RED)
GRAPEVINE 6 7: THOUGHT:ACT:NOAP
GRAPEVINE 68: THOUGHT:REAS:ACHIEVE↑BOT_STATUS[BLOCK(B2)]
= ONTOP TABLE(VAR)
GRAPEVINE 69: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(B1)
GRAPEVINE 70: THOUGHT:REAS:ACHIEVE↑MOVE/HAND(MAX)/

HAND(MAX)/FROM -545 145 -260/TO -75 -450 -410
GRAPEVINE 71: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -545 145 -260/TO -75 -450 -410
GRAPEVINE 72: THOUGHT:REAS:ACHIEVE↑GRASP/HAND(MAX)/
BLOCK(B7)
GRAPEVINE 73: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B7)
GRAPEVINE 74: THOUGHT:REAS:ACHIEVE↑OFFOF/BLOCK(B7)/
BLOCK(B5)
GRAPEVINE 75: THOUGHT:ORD:GETOFFOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#10
GRAPEVINE 76: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B7)/
FROM -75 -450 -410/TO -422 -602 -460
GRAPEVINE 77: THOUGHT:REAS:ACHIEVE↑ONTOP/BLOCK(B7)/
TABLE(TABL1)
GRAPEVINE 78: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑OFFOF/
BLOCK(B7)/BLOCK(B5)
GRAPEVINE 79: THOUGHT:REAS:ACHIEVE↑ONTOP/BLOCK(B5)/
BLOCK(B2)
GRAPEVINE 80: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#11
GRAPEVINE 81: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(B7)
GRAPEVINE 82: THOUGHT:REAS:ACHIEVE↑MOVE/HAND(MAX)/
HAND(MAX)/FROM -422 -608 -460/TO -75 -450 -460
GRAPEVINE 83: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -422 -608 -460/TO -75 -450 -460
GRAPEVINE 84: THOUGHT:REAS:ACHIEVE↑GRASP_STATUS[HAND(MAX)]
- GRASPING BLOCK(B5)
GRAPEVINE 85: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B5)
GRAPEVINE 86: THOUGHT:REAS:ACHIEVE↑MOVE/HAND(MAX)/
BLOCK(B5)/FROM -75 -450 -460,'TO -310 -550 -260
GRAPEVINE 87: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B5)/
FROM -75 -450 -460/TO -310 -550 -260
GRAPEVINE 88: THOUGHT:REAS:ACHIEVE↑ONTOP/BLOCK(B5)/
BLOCK(B2)
GRAPEVINE 89: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ONTOP/
BLOCK(B5)/BLOCK(B2)
GRAPEVINE 90: THOUGHT:REAS:ACHIEVE↑STACKUP/BLOCK(B5)/
BLOCK(B2)
GRAPEVINE 91: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/

```
FIND_SPOT/INFO#12
 GRAPEVINE 92: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
tIAND(MAX)/BLOCK(B5)
 GRAPEVINE 93: THOUGHT:REAS:ACHIEVE↑MOVE/HAND(MAX)/
HAND(MAX)/FROM -310 -550 -260/TO -105 -70 -160
 GRAPEVINE 94: THOUGHT:ACT:↑MOVE/HAND( MAX)/HAND( MAX)/
FROM -310 -550 -260/TO -105 -70 -160
 GRAPEVINE 95: THOUGHT:REAS:ACHIEVE↑GRASP_STATUS[HAND( MAX)]
= GRASPING BLOCK(B6)
 GRAPEVINE 96: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B6)
 GRAPEVINE 97: THOUGHT:REAS:ACHIEVE↑MOVE/HAND(MAX)/
BLOCK(B6)/FROM -105 -70 -160/TO -310 -550 -60
 GRAPEVINE 98: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B6)/
FROM -105 -70 -160/TO -310 -550 -60
 GRAPEVINE 99: THOUGHT:REAS:ACHIEVE↑ONTOP/BLOCK(B6)/
BLOCK(B5)
 GRAPEVINE 100: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ONTOP/
BLOCK(B6)/BLOCK(B5)
 GRAPEVINE 1.01: THOUGHT:REAS:↑ASK/HUMAN(FRIEND)/HAND(MAX)
 GRAPEVINE 102: ANS:HAND(MAX) TO HUMAN(FRIEND)↑YES//
 GRAPEVINE 103: REAS:↑IMAGINE/HAND(MAX)/INFO#13


---------- Question 13 ----------

 GRAPEVINE 104: QUEST:IS_POSSIBLE/HUMAN(FRIEND) TO
HAND(MAX)↑DO/HAND(MAX)/PICKUP BLOCK(COLOR = RED)
 GRAPEVINE 105: ACT:NOAP
 GRAPEVINE 106: REAS:MATCHX/ERROR↑NOT CORRECT/FORM/
COM:↑PICKUP/HAND(MAX)/BLOCK(COLOR = RED)
 GRAPEVINE 107: ANS:HAND( MAX) TO HUMAN(FRIEND)↑YES//
 GRAPEVINE 108: REAS:GRAPEVINE #19↑DONE/BEFORE/


---------- Question 14 ----------

 GRAPEVINE 109: QUEST:IS_POSSIBLE/HUMAN(FRIEND) TO
HAND(MAX)↑DO/HAND(MAX)/STACKUP BLOCK() BLOCK0 AND BLOCK0
 GRAPEVINE 110: ANS:HAND( MAX) TO HUMAN(FRIEND)↑Y ES//
```

GRAPEVINE 111: REAS:GRAPEVINE #103↑THOUGHT_DONE/BEFORE/

---------- Question 15 ----------

GRAPEVINE 112: FACT:HUMAN(FRIEND) TO HAND(MAX)↑EQUAL/
LIKED_STATUS[BLOCK(B8)]/LIKED_BY HUMAN(FRIEND)
GRAPEVINE 113: FACT:HUMAN(FRIEND) TO HAND(MAX)↑EQUAL/
LIKE_STATUS[HUMAN(FRIEND)]/LIKE BLOCK(B8)

Question 16 ----------

GRAPEVINE 114: FACT:HUMAN(FRIEND) TO HAND(MAX)↑EQUAL/
LIKED_STATUS[BLOCK(B1)]/LIKED_BY HUMAN(FRIEND)
GRAPEVINE 115: FACT:HUMAN(FRIEND) TO HAND(MAX)↑EQUAL/
LIKED_STATUS[BLOCK(B3)]/LIKED_BY HUMAN(FRIEND)
GRAPEVINE 116: FACT:HUMAN(FRIEND) TO HAND(MAX)↑EQUAL/
LIKED_STATUS[BLOCK(B4)]/LIKED_BY HUMAN(FRIEND)
GRAPEVINE 117: FACT:HUMAN(FRIEND) TO HAND(MAX)↑EQUAL/
LIKED_STATUS[BLOCK(B7)]/LIKED_BY HUMAN(FRIEND)
GRAFEVINE 118: FACT:HUMAN(FRIEND) TO HAND(MAX)↑EQUAL/
LIKED_STATUS[BLOCK(B8)]/LIKED_BY HUMAN(FRIEND)
GRAPEVINE 119: FACT:HUMAN(FRIEND) TO HAND(MAX)↑EQUAL/
LIKE_STATUS[HUMAN(FRIEND)]/LIKE BLOCK(B8)|LIKE BLOCK(B7)|
LIKE BLOCK(B4)|LIKE BLOCK(B3)|LIKE BLOCK(B1)
GRAPEVINE 120: FACT:HUMAN(FRIEND) TO HAND(MAX)↑EQUAL/
LIKED_STATUS[BLOCK(B4)]/NOT LIKED-BY HUMAN(FRIEND)
GRAFEVINE 121: FACT:HUMAN(FRIEND) TO HAND(MAX)↑EQUAL/
LIKED_STATUS[BLOCK(B5)]/NOT LIKED-BY HUMAN(FRIEND)
GRAPEVINE 122: FACT:HUMAN(FRIEND) TO HAND(MAX)↑EQUAL/
LIKE_STATUS[HUMAN(FRIEND)]/NOT LIKE BLOCK(B5)|NOT LIKE
BLOCK(B4)
GRAPEVINE 123: FACT:HUMAN(FRIEND) TO HAND(MAX)↑EQUAL/
LIKED_STATUS[BOX(BOX1)]/NOT LIKED-BY HUMAN(FRIEND)
GRAPEVINE 124: FACT:HUMAN(FRIEND) TO HAND(MAX)↑EQUAL/
LIKE_STATUS[HUMAN(FRIEND)]/NOT LIKE BOX(BOX1)

```
------------- Question 17 -------------

 GRAPEVINE 125: QUEST:IS_FACT/HUMAN(FRIEND) TO HAND(MAX)
↑EQUAL/LIKE_STATUS[HUMAN(FRIEND)]/LIKE BOX(DEFINITE)
 GRAPEVINE 126: ANS:HAND(MAX) TO HUMAN(FRIEND)↑NO//
 GRAPEVINE 127: REAS:IS_FACT/↑NOT EQUAL/LIKE-STATUS
[HUMAN(FRIEND)]/LIKE BOX( DEFINITE)
 GRAPEVINE 128: QUEST:IS_FACT/HUMAN(FRIEND) TO HAND(MAX)
↑EQUAL/LIKE_STATUS[HUMAN(FRIEND)]/NOT LIKE BOX(DEFINITE)
 GRAPEVINE 129: ANS:HAND( MAX) TO HUMAN(FRIEND)↑YES/
/BOX(BOX1)



------------- Question 18 -------------

 GRAPEVINE 130: QUEST:YES_NO/HUMAN(FRIEND) TO HAND(MAX)
↑EXIST/OBJECT(BOT_STATUS = ONTOP BOX( DEFINITE)|LIKED_STATUS
= LIKED_BY HUMAN( FRIEND))/
 GRAPEVINE 133 : ANS:HAND(MAX) TO HUMAN(FRIEND)↑EXIST/
BLOCK(VAR B1 B8)/2



------------- Question 19 -------------

 GRAPEVINE 132: QUEST:WHAT_IS/HUMAN(FRIEND) TO HAND(MAX)↑IS/
LIKED_STATUS/BLOCK(COLOR = GREEN|KINDOF = CUBE|LOCASHUN
IN-FRONT-OF BLOCK(COLOR = GREEN))
. GRAPEVINE 133: ANS:HAND(MAX) TO HUMAN(FRIEND)↑IS/
LIKED-STATUS/NOT LIKED-BY HUMAN(FRIEND)



------------- Question 20 -------------

 GRAPEVINE 134: COM:HUMAN(FRIEND) TO HAND(MAX)↑STACKUP/
HAND(MAX)/BLOCK(COLOR = RED) BLOCK(COLOR = RED) AND
EITHER BLOCK(COLOR = GREEN) OR BLOCK(KINDOF = PYRAMID)
 GRAPEVINE 135: ACT:NOAP
 GRAPEVINE 136: REAS:ACHIEVE↑BOT_STATUS[BLOCK(B2)]
= ONTOP TABLE(VAR)
```

GRAPEVINE 137: ORD:GETONTOPOF TO FIND_SPOT↑FIND/FIND_SPOT/
INFO#14
GRAPEVINE 138: ACT:CAUSED_BY HAND(MAX)↑RELEASE/HAND(MAX)/
BLOCK(B1)
GRAPEVINE 139: REAS:ACHIEVE↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -545 145 -260/TO -545 -255 -360
GRAPEVINE 140: ACT:↑MOVE/HAND(MAX)/HAND(MAX)/FROM
-545 145 -260/TO -545 -255 -360
GRAPEVINE 141: REAS:ACHIEVE↑GRASP_STATUS[HAND(MAX)]
= GRASPING BLOCK(B3)
GRAPEVINE 142: ACT:CAUSED,BY HAND(MAX)↑GRASP/HAND(MAX)/
BLOCK(B3)
GRAPEVINE 143: REAS:ACHIEVE↑MOVE/HAND(MAX)/BLOCK(B3)/
FROM -545 -255 -360/TO -310 -550 -160
GRAPEVINE 144: ACT:↑MOVE/HAND(MAX)/BLOCK(B3)/FROM
-545 -255 -360/TO -310 -550 -160
GRAPEVINE 145: REAS:ACHIEVE↑ONTOP/BLOCK(B3)/BLOCK(B2)
GRAPEVINE 146: ACT:CAUSED,BY HAND(MAX)↑ONTOP/BLOCK(B3)/
BLOCK(B2)
GRAPEVINE 147: REAS:ACHIEVE↑STACKUP/BLOCK(B3)/BLOCK(B2)
GRAPEVINE 148: ACT:CAUSED,BY HAND(MAX)↑RELEASE/HAND(MAX)/
BLOCK(B3)
GRAPEVINE 149: REAS:ACHIEVE↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -310 -550 -160/TO -75 -450 -410
GRAPEVINE 150: ACT:↑MOVE/HAND(MAX)/HAND(MAX)/FROM
-310 -550 -160/TO -75 -450 -410
GRAPEVINE 151: REAS:ACHIEVE↑GRASP/HAND(MAX)/BLOCK(B7)
GRAPEVINE 152: ACT:CAUSED_BY HAND(MAX)↑GRASP/HAND(MAX)/
BLOCK(B7)
GRAPEVINE 153: REAS:ACHIEVE↑OFFOF/BLOCK(B7)/BLOCK(B5)
GRAPEVINE 154: ORD:GETOFFOF TO FIND_SPOT↑FIND/FIND_SPOT/
INFO#15
GRAPEVINE 155: ACT:↑MOVE/HAND(MAX)/BLOCK(B7)/FROM
-75 -450 -410/TO -422 -608 -460
GRAPEVINE 156: REAS:ACHIEVE↑ONTOP/BLOCK(B7)/TABLE(TABL1)
GRAPEVINE 157: ACT:CAUSED,BY HAND(MAX)↑OFFOF/BLOCK(B7)/
BLOCK(B5)
GRAPEVINE 158: REAS:ACHIEVE↑ONTOP/BLOCK(B5)/BLOCK(B3)
GRAPEVINE 159: ORD:GETONTOPOF TO FIND_SPOT↑FIND/FIND_SPOT/
INFO#16
GRAPEVINE 160: ACT:CAUSED,BY HAND(MAX)↑RELEASE/HAND(MAX)/
BLOCK(B7)

GRAPEVINE 161: REAS:ACHIEVE↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -422 -608 -460/TO -75 -450 -460
GRAPEVINE 162: ACT:↑MOVE/HAND(MAX)/HAND(MAX)/FROM
-422 -608 -460/TO -75 -450 -460
GRAPEVINE 163: REAS:ACHIEVE↑GRASP_STATUS[HAND(MAX)]
= GRASPING BLOCK(B5)
GRAPEVINE 164: ACT:CAUSED_BY HAND(MAX)↑GRASP/HAND(MAX)/
BLOCK(B5)
GRAPEVINE 165: REAS:ACHIEVE↑MOVE/HAND(MAX)/BLOCK(B5)/
FROM -75 -450 -460/TO -310 -550 -110
GRAPEVINE 166: ACT:↑MOVE/HAND(MAX)/BLOCK(B5)/FROM
-75 -450 -460/TO -310 -550 -110
GRAPEVINE 167: REAS:ACHIEVE↑ONTOP/BLOCK(B5)/BLOCK(B3)
GRAPEVINE 168: ACT:CAUSED,BY HAND(MAX)↑ONTOP/BLOCK(B5)/
BLOCK(B3)
GRAPEVINE 169: REAS:↑ASK/HUMAN(FRIEND)/HAND(MAX)
GRAPEVINE 170: ANS:HAND(MAX) TO HUMAN(FRIEND)↑DID/
HAND(MAX)/INFO#17

---------- Question 21 ----------

GRAPEVINE 171: COM:HUMAN(FRIEND) TO HAND(MAX)↑FIND/
HAND(MAX)/BLOCK(KINDOF = CUBE|BOT_STATUS = ONTOP T-
ABLE(DEFINITE))
GRAPEVINE 172: ANS:HAND(MAX) TO HUMAN(FRIEND)↑FIND/
HAND(MAX)/BLOCK(B4)

---------- Question 22 ----------

GRAPEVINE 173: QUEST:HAVE_IN_COMMON/HUMAN(FRIEND)
TO HAND(MAX)↑HAVE_IN_COMMON/BLOCK(B6)/BLOCK(B5)
GRAPEVINE 174: FACT:↑HAVE_IN_COMMON/BLOCK(B4)/BLOCK(B3)/
DESCRIPTION|DIMENSIONS
GRAPEVINE 175: ANS:↑HAVE_IN_COMMON/BLOCK(B6)/BLOCK(B5)/
CATEGORY |COLOR|SIZE|XLENGTH|YWIDTH|~BOT_STATUS|TOP_STATUS

---------- Question 23 ----------

GRAPEVINE I. 76: QUEST:HAVE_IN_COMMON/HUMAN(FRIEND)
TO HAND(MAX)↑HAVE_IN_COMMON/BLOCK(B3)/BLOCK(B4)
GRAPEVINE 177: FACT:↑HAVE_IN_COMMON/BLOCK(B2)/TABLE(TABL1)/
XCMS
GRAPEVINE I. 78: FACT:↑HAVE_IN_COMMON/BLOCK(B5)/BLOCK(B6)/
CATEGORY |COLOR|SIZE|XLENGTH|YWIDTH|TOP_STATUS
GRAPEVINE 179: ANS:↑HAVE_IN_COMMON/BLOCK(B3)/BLOCK(B4)/
DESCRIPTION|DIMENSIONS|~TOP_STATUS

---------- Question 24 ----------

GRAPEVINE 3.80: QUEST:YES_NO/HUMAN(FRIEND) TO HAND(MAX)
↑EXIST/BLOCK(SIZE = BIG|KINDOF = PARALLELEPIPED|XCOORD
< XCOORD[∃ BLOCK(KINDOF = PYRAMID)])/
GRAPEVINE 1.81: ANS:HAND(MAX) TO HUMAN(FRIEND)↑EXIST/
BLOCK(VAR B1 B2 B3)/3

------- ------- Question 25 ----------

GRAPEVINE 182: COM:HUMAN(FRIEND) TO HAND(MAX)↑ONTOP/
BLOCK(SIZE = SMALL|KINDOF = PARALLELEPIPED)/BLOCK(COLOR
= GREENITOP,STATUS = HOLDING BLOCK(KINDOF = PYRAMID))
GRAPEVINE 183: ORD:GETONTOPOF TO FIND_SPOT↑FIND/FIND_SPOT/
INFO::18
GRAPEVINE 1.84: ACT:↑MOVE/HAND(MAX)/BLOCK(B5)/FROM
-310 -550 -110/TO -55 -155 -310
GRAPEVINE 185: REAS:ACHIEVE↑ONTOP/BLOCK(B5)/BLOCK(B4)
GRAPEVINE 186: ACT:CAUSED_BY HAND(MAX)↑ONTOP/BLOCK(B5)/
BLOCK(B4)
GRAPEVINE 187: REAS:↑ASK/HUMAN(FRIEND)/HAND(MAX)
GRAPEVINE 1.88: ANS:HAND(MAX) TO HUMAN(FRIEND)↑DID/
HAND(MAX)/INFO#19

---------- Question 26 ----------

GRAPEVINE 189: COM:HUMAN(FRIEND) TO HAND(MAX)↑ONTOP/
BLOCK(KINDOF = PYRAMID|HEIGHT ≤ HEIGHT[∀ BLOCK(KINDOF
= PYRAMID)])/BLOCK(ACT:↑ONTOP/X/ ← 188 3)
GRAPEVINE I. 90: ORD:GETONTOPOF TO FIND_SPOT↑FIND/FIND_SPOT/
INFO#20
GRAPEVINE 191: ACT:CAUSED,BY HAND(MAX)↑RELEASE/HAND(MAX)/
BLOCK(B5)
GRAFEVINE 1.92: REAS:ACHIEVE↑MOVE/HAND(MAX)/HAND(MAX)/
FROM --55 -155 -310/TO -422 -601 -460
GRAPEVINE 193: ACT:↑MOVE/HAND(MAX)/HAND(MAX)/FROM
-55 -155 -310/TO -422 -608 -460
GRAPEVINE 194: REAS:ACHIEVE↑GRASP_STATUS[HAND(MAX)]
= GRASPING BLOCK(B7)
GRAPEVINE 195: ACTCAUSED,BY HAND(MAX)↑GRASP/HAND(MAX)/
BLOCK(B7)
GRAPEVINE 196: REAS:ACHIEVE↑MOVE/HAND(MAX)/BLOCK(B7)/
FROM -422 -608 -460/TO -55 -155 -260
GRAPEVINE 197: ACT:↑MOVE/HAND(MAX)/BLOCK(B7)/FROM
-422 -608 -460/TO -55 -155 -260
GRAPEVINE 1.98: REAS:ACHIEVE↑ONTOP/BLOCK(B7)/BLOCK(B5)
GRAPEVINE 199: ACT:CAUSED,BY HAND(MAX)↑ONTOP/BLOCK(B7)/
BLOCK(B5)
GRAPEVINE 200: REAS:↑ASK/HUMAN(FRIEND)/HAND(MAX)
GRAPEVINE 201: ANS:HAND(MAX) TO HUMAN(FRIEND)↑DID/
HAND(MAX)/INFO#21

---------- Question 27 ----------

GRAPEVINE 202: QUEST:YES_NO/HUMAN(FRIEND) TO HAND(MAX)
↑EXIST/OBJECT(TOP_STATUS = HOLDING BLOCK(KINDOF = PYRAMID|
HEIGHT ≥ HEIGHT[∀ BLOCK(KINDOF = PYRAMID)]))/
GRAPEVINE 203: ANS:HAND(MAX) TO HUM AN(FRIEND)? EXIST/
BLOCK(B4)/1
GRAPEVINE 204: QUEST:YES_NO/HUMAN(FRIEND) TO HAND(MAX)
↑EXIST/OBJECT(COLOR = GREENIBOT,STATUS = ONTOP
OBJECT(BOT_STATUS = ONTOP OBJECT(ANS:↑EXIST/X/ → 201 3))|
HEIGHT ≤ HEIGHT[∀ OBJECT(BOT_STATUS = ONTOP OBJECT(ANS:↑
EXIST/X/ → 201 3))])/
GRAPEVINE 205: ANS:HAND(MAX) TO HUMAN(FRIEND)↑EXIST/

BLOCK(B7)/1
GRAPEVINE 206: QUEST:YES_NO/HUMAN(FRIEND) TO HAND(MAX)
↑EXIST/OBJECT(COLOR = GREEN|BOT_STATUS = ONTOP OBJECT(HEIGHT
≤ HEIGHT[∀ OBJECT(BOT_STATUS = ONTOP OBJECT(TOP_STATUS
= HOLDING BLOCK(KINDOF = PYRAMID|HEIGHT ≥ HEIGHT[∀
BLOCK(KINDOF = PYRAMID)])))])/
GRAPEVINE 207: ANS:HAND(MAX) TO HUMAN(FRIEND)↑EXIST/
BLOCK(B7)/1

---------- Question 28 ----------

GRAPEVINE 208: QUEST:WHAT_IS/HUMAN(FRIEND) TO HAND(MAX)↑IS/
COLOR/OBJECT(TOP_STATUS = HOLDING OBJECT(ANS:↑EXIST/
X/ ← 207 2))
GRAPEVINE 209: ANS:HAND(MAX) TO HUMAN(FRIEND)↑IS/COLOR/RED

---------- Question 29 ----------

GRAPEVINE 210: QUEST:HOW_MANY/HUMAN(FRIEND) TO HAND(MAX)
↑EXIST/OBJECT(LOCASHUN DIRECTLY-ABOVE BLOCK(COLOR = GREEN|
KINDOF = CUBE))/
GRAPEVINE 211: ANS:HAND(MAX) TO HUMAN(FRIEND)↑EXIST/
BLOCK(VAR B5 B6 B7)/3

---------- Question 30 ----------

GRAPEVINE 212: QUEST:YES_NO/HUMAN(FRIEND) TO HAND(MAX)
↑EXIST/BLOCK(KINDOF = PYRAMID|ACT:↑GRASP/HAND(MAX)/
X ← COM:↑ONTOP/BLOCK(COLOR = GREEN|KINDOF = PYRAMID)/
BLOCK(SIZE = SMALL|KINDOF = CUBE)← ∞)/
GRAPEVINE 213: ACT:NOAP
GRAPEVINE 214: REAS:INSTANCES/↑NOT EXIST/BLOCK/BLOCK(KINDOF
= PYRAMID|PNUMBER = VAR B5)
GRAPEVINE 215: ANS:HAND(MAX) TO HUMAN(FRIEND)↑EXIST/
BLOCK(B7)/1

---------- Question 31 ----------

GRAPEVINE 216: QUEST:WHEN_DID/HUMAN(FRIEND) TO HAND(MAX)↑DO/
HAND(MAX)/PICKUP BLOCK(ANS:↑EXIST/X/ ← 215 ∞)
GRAPEVINE 217: ANS:HAND(MAX) TO HUMAN(FRIEND)↑DID/
HAND(MAX)/#195/WHILE#201

---------- Question 32 ----------

GRAPEVINE 218: QUEST:WHEN_DID/HUMAN(FRIEND) TO HAND(MAX)↑DO/
HAND(MAX)/ACT:↑GRASP/HAND(MAX)/BLOCK(ANS:↑EXIST/X/
← 21 7 ∞) → FIRST ∞
GRAPEVINE 219: ANS:HAND(MAX) TO HUMAN(FRIEND)↑ DID/
HAND(MAX)/#152/WHILE#170

---------- Question 33 ----------

GRAPEVINE 220: QUEST:WHY_DID/HUMAN(FRIEND) TO HAND(MAX)↑DO/
HAND(MAX)/[ACT:↑// → 152 2]
GRAPEVINE 221: ANS:HAND(MAX) TO HUMAN(FRIEND)↑ ACTED BECAUSE/
HAND(MAX)/#153

---------- Question 34 ----------

GRAPEVINE 222: QUEST:WHY_DID/HUMAN(FRIEND) TO HAND(MAX)↑DO/
HAND(MAX)/[ACT:↑OFFOF//BLOCK(KINDOF = CUBE) ↔1534]
GRAPEVINE 223: ANS:HAND(MAX) TO HUMAN(FRIEND)↑ACTED BECAUSE/
HAND(MAX)/#158

---------- Question 35 ----------

GRAPEVINE 224: QUEST:WHY_DID/HUMAN(FRIEND) TO HAND(MAX)↑DO/

HAND(MAX)/[ACT:↑// → 158 2]
GRAPEVINE 225: ANS:HAND(MAX) TO HUMAN(FRIEND)↑ACTED BECAUSE/
HAND(MAX)/#161

---------- Question 3 6 ----------

GRAPEVINE 226: QUEST:WHY_DID/HUMAN(FRIEND) TO HAND(MAX)↑DO/
HAND(MAX)/[ACT:↑// → 161 2 3
GRAPEVINE 227: ANS:HAND(MAX) TO HUMAN(FRIEND)↑ACTED BECAUSE/
HAND(MAX)/#163

---------- Question 3 7 ----------

GRAPEVINE 228: QUEST:WHY_DID/HUMAN(FRIEND) TO HAND(MAX)↑DO/
HAND(MAX)/[ACT:↑// → 163 2]
GRAPEVINE 229: ANS:HAND(MAX) TO HUMAN(FRIEND)↑ACTED BECAUSE/
HAND(MAX)/#165

---------- Question 3 8 ----------

GRAPEVINE 230: QUEST:WHY_DID/HUMAN(FRIEND) TO HAND(MAX)↑DO/
HAND(MAX)/[ACT:↑// → 165 2]
GRAPEVINE 231: ANS:HAND(MAX) TO HUMAN(FRIEND)↑ACTED BECAUSE/
HAND(MAX)/#167

---------- Question 39 ----------

GRAPEVINE 232: QUEST:WHY_DID/HUMAN(FRIEND) TO HAND(MAX)↑DO/
HAND(MAX)/[ACT:↑// → 167 2]
GRAPEVINE 233: ANS:HAND(MAX) TO HUMAN(FRIEND)↑ACTED BECAUSE/
HAND(MAX)/#169

---------- Question 40 ----------

GRAPEVINE 234: QUEST:HOW_DID/HUMAN(FRIEND) TO HAND(MAX)↑DO/
HAND(MAX)/[COM:↑// ← 169 ∞]
GRAPEVINE 235: ANS:HAND(MAX) TO HUMAN(FRIEND)↑DID/
HAND(MAX)/FROM#134/TO#170

---------- Question 41 ----------

GRAPEVINE 236:QUEST:HOW_MANY/HUMAN(FRIEND) TO HAND(MAX)
↑EXIST/OBJECT()/
GRAPEVINE 237: ANS:HAND(MAX) TO HUMAN(FRIEND)↑EXIST/

---------- Question 42 ----------

BLOCK(VAR B3 B5 B7)/3
GRAPEVINE 238: QUEST:HOW_MANY/HUMAN(FRIEND) TO HAND(MAX)
↑EXIST/OBJECT(COLOR = RED)/
GRAPEVINE 239: ANS:HAND(MAX) TO HUMAN(FRIEND)↑EXIST/
BLOCK(B2)/1

---------- Question 43 ------L----w---

GRAPEVINE 240: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B7)/
FROM -55 -1.55 -260/TO -422 -608 -460
GRAPEVINE 241: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 242: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#22
GRAPEVINE 243: THOUGHT:ACT:NOAP
GRAPEVINE 244: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B7)/FROM -422 -608 -460/TO -422 -608 -460
GRAPEVINE 245: THOUGHT:ACT:CAUSED,BY HAND(MAX)↑ONTOP/
BLOCK(B7)/TABLE(TABL1)
GRAPEVINE 246: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B7)
GRAPEVINE 247: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/

HAND(MAX)/BLOCK(B7)
GRAPEVINE 248: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
GRAPEVINE 249: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -422 -608 -460/TO -55 -155 -310
GRAPEVINE 250: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 251: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B5)
GRAPEVINE 252: THOUGHT:REAS:↑ASK/AT_TIME/GRASP
GRAPEVINE 253: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B5)/
FROM -55 -155 -310/TO -310 -550 -110
GRAPEVINE 254: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 255: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B5)/
FROM -310 -550 -110/TO -75 -450 -460
GRAPEVINE 256: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 257: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#23
GRAPEVINE 258: THOUGHT:ACT:NOAP
GRAPEVINE 259: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B5)/FROM -75 -450 -460/TO -75 -450 -460
GRAPEVINE 260: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ONTOP/
BLOCK(B5)/TABLE(TABL1)
GRAPEVINE 261: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B5)
GRAPEVINE 262: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(B5)
GRAPEVINE 263: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
GRAPEVINE 264: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -75 -450 -460/TO -422 -608 -460
GRAPEVINE 265: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 266: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B7)
GRAPEVINE 36 7: THOUGHT:REAS:↑ ASK/AT-TIME/GRASP
GRAPEVINE 265: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B7)/
FROM -422 -608 -460/TO -75 -450 -410
GRAPEVINE 269: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 270: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#24
GRAPEVINE 271 : THOUGHT:ACT:NOAP
GRAPEVINE 272: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B7)/FROM -75 -450 -410/TO -75 -450 -410
GRAPEVINE 273: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ONTOP/
BLOCK(B7)/BLOCK(B5)

GRAPEVINE 274: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B7)
GRAPEVINE 275: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(B7)
GRAPEVINE 276: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
GRAPEVINE 2 77: THOUGHT:COM:HUMAN(FRIEND) TO HAND(MAX)↑FIND/
HAND(MAX)/OBJECT(BOT_STATUS = ONTOP BLOCK(COLOR = RED|
KINDOF = CUBE))
GRAPEVINE 278: ANS:HAND(MAX) TO HUMAN(FRIEND)↑FIND/
HAND(MAX)/BLOCK(B7)

---------- Question 4 4 e--v----------

GRAPEVINE 279: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B7)/
FROM -55 -155 -260/TO -422 -608 -460
GRAPEVINE 280: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 281: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#25
GRAPEVINE 252: THOUGHT:ACT:NOAP
GRAPEVINE 283: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B7)/FROM -422 -608 -460/TO -422 -608 -460
GRAPEVINE 284: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ONTOP/
BLOCK(B7)/TABLE(TABL1)
GRAPEVINE 285: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B7)
GRAPEVINE 286: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(B7)
GRAPEVINE 287: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
GRAPEVINE 288: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -422 -608 -460/TO -55 -155 -310
GRAPEVINE 289: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 290: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B5)
GRAPEVINE 291:THOUGHT:REAS:↑ASK/AT_TIME/GRASP
GRAPEVINE 292: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B5)/
FROM -55 -155 -310/TO -310 -550 -110
GRAPEVINE 293: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 294: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B5)/
FROM -3113 -550 -110/TO -75 -450 -460
GRAPEVINE 295: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO

GRAPEVINE 296: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#26
GRAPEVINE 297: THOUGHT:ACT:NOAP
GRAPEVINE 298: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B5)/FROM -75 -450 -460/TO -75 -450 -460
GRAPEVINE 299: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ONTOP/
BLOCK(B5)/TABLE(TABL1)
GRAPEVINE 300: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B5)
GRAPEVINE 301: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
tIAND(MAX)/BLOCK(B5)
GRAPEVINE 302: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
GRAPEVINE 303: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -75 -450 -460/TO -422 -608 -460
GRAPEVINE 304: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 305: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B7)
GRAPEVINE 306: THOUGHT:REAS:↑ASK/AT_TIME/GRASP
GRAPEVINE 307: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B7)/
FROM -422 -608 -460/TO -75 -450 -410
GRAFEVINE 308: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 309: THOUGHT:ORD:GETONTOPOF TOFIND_SPOT↑FIND/
FIND_SPOT/INFO#27
GRAFEVINE 310:THOUGHT:ACT:NOAP
GRAPEVINE 311:THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B7)/FROM -75 -450 -410/TO -75 -450 -410
GRAPEVINE 312: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ONTOP/
BLOCK(B7)/BLOCK(B5)
GRAPEVINE 313: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B7)
GRAPEVINE 314: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(B7)
GRAPEVINE 315: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
GRAPEVINE 316: QUEST:HOW_MANY/HUMAN(FRIEND) TO HAND(MAX)
↑EXIST/BLOCK(LOCASHUN LEFT-OF BOX(DEFINITE)|KINDOF
= PARALLELEPIPED)/
GRAPEVINE 317: ANS:HAND(MAX) TO HUMAN(FRIEND)↑EXIST/
BLOCK(VAR B2 B3 B4 B5)/4

---------- Question 45 ----------

GRAPEVINE 318: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B7)/
FROM -55 -155 -260/TO -422 -608 -460
GRAPEVINE 319: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 320: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#28
GRAPEVINE 321: THOUGHT:ACT:NOAP
GRAPEVINE 322: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B7)/FROM -422 -602 -460/TO -422 -608 -460
GRAPEVINE 323: THOUGHT:ACT:CAUSED,BY HAND(MAX)↑ONTOP/
BLOCK(B7)/TABLE(TABL1)
GRAPEVINE 324: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B7)
GRAPEVINE 325: THOUGHT:ACT:CAUSED,BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(B7)
GRAPEVINE 326: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
GRAPEVINE 327: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -422 -608 -460/TO -55 -155 -310
GRAPEVINE 328: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 329: THOUGHT:ACT:CAUSED,BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B5)
GRAPEVINE 330: THOUGHT:REAS:↑ASK/AT_TIME/GRASP
GRAPEVINE 331: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B5)/
FROM -55 -155 -310/TO -310 -550 -110
GRAPEVINE 332: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 333: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B5)/
FROM -310 -550 -110/TO -75 -450 -460
GRAPEVINE 334: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 335: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#29
GRAPEVINE 336: THOUGHT:ACT:NOAP
GRAPEVINE 337: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B5)/FROM -75 -450 -460/TO -75 -450 -460
GRAPEVINE 338: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ONTOP/
BLOCK(B5)/TABLE(TABL1)
GRAPEVINE 339: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B5)
GRAPEVINE 340: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(B5)
GRAPEVINE 341: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
GRAPEVINE 342: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -75 -450 -460/TO -422 -608 -460

GRAPEVINE 343: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 344: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B7)
GRAPEVINE 345: THOUGHT:REAS:↑ASK/AT_TIME/GRASP
GRAPEVINE 346: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B7)/
FROM -422 -608 -460/TO -75 -450 -410
GRAPEVINE 347: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 348: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#30
GRAPEVINE 349: THOUGHT:ACT:NOAP
GRAPEVINE 350: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B7)/FROM -75 -450 -410/TO -75 -450 -410
GRAPEVINE 351: THOUGHT:ACT:CAUSED,BY HAND(MAX)↑ONTOP/
BLOCK(B7)/BLOCK(B5)
GRAPEVINE 352: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B7)
GRAPEVINE 353: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(B7)
GRAPEVINE 354: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
GRAPEVINE 355: QUEST:HOW_MANY/HUMAN(FRIEND) TO HAND(MAX)
↑EXIST/OBJECT(LOCASHUN DIRECTLY-ABOVE BLOCK(COLOR = GREEN|
KINDOF = CUBE))/
GRAPEVINE 356: ANS:HAND(MAX) TO HUMAN(FRIEND)↑EXIST/
BLOCK(B6)/1

---------- Question 46 ----------

GRAPEVINE 357: COM:HUMAN(FRIEND) TO HAND(MAX)↑ONTOP/
BLOCK(COLOR = BLUE|KINDOF = PYRAMID)/BLOCK(BOT_STATUS
= ONTOP BOX(DEFINITE))
GRAPEVINE 358: ORD:GETONTOPOF TO FIND_SPOT↑FIND/FIND_SPOT/
INFO#31
GRAPEVINE 359: ACT:CAUSED_BY HAND(MAX)↑RELEASE/HAND(MAX)/
BLOCK(B7)
GRAPEVINE 360: REAS:ACHIEVE↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -55 -155 -260/TO -470 165 -410
GRAPEVINE 361: ACT:↑MOVE/HAND(MAX)/HAND(MAX)/FROM
-55 -1 55 -260/TO -470 165 -410
GRAPEVINE 362: REAS:ACHIEVE↑GRASP_STATUS[HAND(MAX)]
= GRASPING BLOCK(B8)

GRAPEVINE 363: ACT:CAUSED_BY HAND(MAX)↑GRASP/HAND(MAX)/
BLOCK(B8).
GRAPEVINE 364: REAS:ACHIEVE↑MOVE/HAND(MAX)/BLOCK(B8)/
FROM -470 165 -410/TO -545 145 -160
GRAPEVINE 365: ACT:↑MOVE/HAND(MAX)/BLOCK(B8)/FROM
-470 1.65 -410/TO -545 145 -160
GRAPEVINE 366: REAS:ACHIEVE↑ONTOP/BLOCK(B8)/BLOCK(B1)
GRAPEVINE 367: ACT:CAUSED,BY HAND(MAX)↑ONTOP/BLOCK(B8)/
BLOCK(B1)
GRAPEVINE 368: REAS:↑ASK/HUMAN(FRIEND)/HAND(MAX)
GRAPEVINE 369: ANS:HAND(MAX) TO HUMAN(FRIEND)↑DID/
HAND(MAX)/INFO#32

---------- Question 47 ----------

GRAPEVINE 370: QUEST:YES_NO/HUMAN(FRIEND) TO HAND(MAX)
↑EXIST/OBJECT(HEIGHT > HEIGHT[∀ BLOCK(KINDOF = PYRAMID)]|
BOT_STATUS = ONTOP OBJECT(YWIDTH > YWIDTH[ITSELF]))/
GRAPEVINE 371: ANS:HAND(MAX) TO HUMAN(FRIEND)↑EXIST/
BLOCK(B1)/1

---------- Question 48 ----------

GRAPEVINE 372: QUEST:YES_NO/HUMAN(FRIEND) TO HAND(MAX)
↑EXIST/STEEPLE()/
GRAPEVINE 373: ACT:NOAP
GRAPEVINE 374: REAS:WHATISIT/↑NOT EXIST/COMPOSITE_OBJECT/
STEEPLE

---------- Question 49 ----------

GRAPEVINE 375: FACT:HUMAN(FRIEND) TO HAND(MAX)↑EQUAL/
STEEPLE()/BLOCK(COLOR=GREEN|KINDOF = CUBEITOP,STATUS
= HOLDING BLOCK(COLOR = GREEN|KINDOF = CUBEITOP,STATUS
= HOLDING BLOCK(COLOR = RED|KINDOF = PYRAMID)))

-- ----- ------- Question 50 ----------

GRAPEVINE 376: QUEST:YES_NO/HUMAN(FRIEND) TO HAND(MAX)
↑EXIST/STEEPLE()/
GRAPEVINE 3 77: ACT:NOAP
GRAPEVINE 378: REAS:INSTANCES/↑NOT EXIST/BLOCK/BLOCK(COLOR
= GREEN|KINDOF = CUBE|TOP_STATUS = HOLDING BLOCK(COLOR
= GREEN|KINDOF = CUBEITOP,STATUS = HOLDING BLOCK(COLOR = RED|
KINDOF = PYRAMID)))
GRAPEVINE 379: ANS:HAND(MAX) TO HUMAN(FRIEND)↑NOT EXIST/
BLOCK(COLOR = GREEN|KINDOF = CUBE/TOP-STATUS = HOLDING
BLOCK(COLOR = GREENIKINDOF = CUBEITOP,STATUS = HOLDING
BLOCK(COLOR = RED|KINDOF = PYRAMID))))/

---------- Question 51 ----------

GRAPEVINE 350: COM:HUMAN(FRIEND) TO HAND(MAX)↑BUILD/
HAND(MAX)/STEEPLE()
GRAPEVINE 381: ORD:GETONTOPOF TO FIND_SPOT↑FIND/FIND_SPOT/
INFO//34
GRAPEVINE 382: ACT:NOAP
GRAPEVINE 383: REAS:FIND_SPOT/CANNOT_BE_DONE↑IS/AREA[(FREE)
OF BLOCK(B4)]/TOOSMALL
GRAPEVINE 384: ORD:GETONTOPOF TO FIND_SPOT↑FIND/FIND_SPOT/
INFO//35
GRAPEVINE 385: ACT:CAUSED,BY HAND(MAX)↑RELEASE/HAND(MAX)/
BLOCK(B8)
GRAPEVINE 386: REAS:ACHIEVE↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -545 145 -160/TO -55 -155 -260
GRAPEVINE 387: ACT:↑MOVE/HAND(MAX)/HAND(MAX)/FROM
-545 145 -160/TO -55 -155 -260
GRAPEVINE 388: REAS:ACHIEVE↑GRASP/HAND(MAX)/BLOCK(B7)
GRAPEVINE 389: ACT:CAUSED_BY HAND(MAX)↑GRASP/HAND(MAX)/
BLOCK(B7)
GRAPEVINE 390: REAS:ACHIEVE↑OFFOF/BLOCK(B7)/BLOCK(B5)
GRAPEVINE 391: ORD:GETOFFOF TO FIND_SPOT↑FIND/FIND_SPOT/
INFO//36
GRAPEVINE 392: ACT:↑MOVE/HAND(MAX)/BLOCK(B7)/FROM

-55 -155 -260/TO -422 -608 -460
GRAPEVINE 393: REAS:ACHIEVE↑ONTOP/BLOCK(B7)/TABLE(TABL1)
GRAPEVINE 394: ACT:CAUSED,BY HAND(MAX)↑OFFOF/BLOCK(B7)/
BLOCK(B5)
GRAPEVINE 395: REAS:ACHIEVE↑TOP_STATUS[BLOCK(B5)] = CLEAR
GRAPEVINE 396: ACT:CAUSED_BY HAND(MAX)↑RELEASE/HAND(MAX)/
BLOCK(B7)
GRAPEVINE '397: REAS:ACHIEVE↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -422 -608 -460/TO -55 -155 -310
GRAPEVINE 398: ACT:↑MOVE/HAND(MAX)/HAND(MAX)/FROM
-422 -608 -460/TO -55 -155 -310
GRAPEVINE 399: REAS:ACHIEVE↑GRASP/HAND(MAX)/BLOCK(B5)
GRAPEVINE 400: ACT:CAUSED_BY HAND(MAX)↑GRASP/HAND(MAX)/
BLOCK(B5)
GRAPEVINE 401: REAS:ACHIEVE↑OFFOF/BLOCK(B5)/BLOCK(B4)
GRAPEVINE 402: ORD:GETOFFOF TO FIND_SPOT↑FIND/FIND_SPOT/
INFO//37
GRAPEVINE 403: ACT:↑MOVE/HAND(MAX)/BLOCK(B5)/FROM
-55 -1 55 -310/TO -472 -608 -460
GRAPEVINE 404: REAS:ACHIEVE↑ONTOP/BLOCK(B5)/TABLE(TABL1)
GRAPEVINE 405: ACT:CAUSED_BY HAND(MAX)↑OFFOF/BLOCK(B5)/
BLOCK(B4)
GRAPEVINE 406: REAS:ACHIEVE↑TOP_STATUS[BLOCK(B4)] = CLEAR
GRAPEVINE 407: ACT:CAUSED_BY HAND(MAX)↑RELEASE/HAND(MAX)/
BLOCK(B5)
GRAPEVINE 408: REAS:ACHIEVE↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -472 -608 -460/TO -105 -70 -160
GRAPEVINE 409: ACT:↑MOVE/HAND(MAX)/HAND(MAX)/FROM
-472 -608 -460/TO -105 -70 -160
GRAPEVINE 410: REAS:ACHIEVE↑GRASP/HAND(MAX)/BLOCK(B6)
GRAPEVINE 411: ACT:CAUSED_BY HAND(MAX)↑GRASP/HAND(MAX)/
BLOCK(B6)
GRAPEVINE 412: REAS:ACHIEVE↑OFFOF/BLOCK(B6)/BLOCK(B4)
GRAPEVINE 413: ORD:GETOFFOF TO FIND_SPOT↑FIND/FIND_SPOT/
INFO//38
GRAPEVINE 414: ACT:↑MOVE/HAND(MAX)/BLOCK(B6)/FROM
-105 -70 -160/TO -522 -608 -310
GRAPEVINE 415: REAS:ACHIEVE↑ONTOP/BLOCK(B6)/TABLE(TABL1)
GRAPEVINE 416: ACT:CAUSED_BY HAND(MAX)↑OFFOF/BLOCK(B6)/
BLOCK(B4)
GRAPEVINE 417: REAS:ACHIEVE↑TOP_STATUS[BLOCK(B4)] = CLEAR
GRAPEVINE 418: ACT:CAUSED_BY HAND(MAX)↑RELEASE/HAND(MAX)/

```
BLOCK(B6)
 GRAPEVINE 419: REAS:ACHIEVE↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -522 -608 -310/TO -310 -550 -160
 GRAPEVINE 420: ACT:↑MOVE/HAND(MAX)/HAND(MAX)/FROM
-522 -608 -310/TO -310 -550 -160
 GRAPEVINE 421: REAS:ACHIEVE↑GRASP_STATUS[HAND(MAX)]
= GRASPING BLOCK(B3)
 GRAPEVINE 422: ACT:CAUSED,BY HAND(MAX)↑GRASP/HAND(MAX)/
BLOCK(B3)
 GRAPEVINE 423: REAS:ACHIEVE↑MOVE/HAND(MAX)/BLOCK(B3)/
FROM -310 -550 -160/TO -75 -105 -210
 GRAPEVINE 424: ACT:↑MOVE/HAND(MAX)/BLOCK(B3)/FROM
-310 -550 -160/TO -75 -105 -210
 GRAPEVINE 425: REAS:ACHIEVE↑ONTOP/BLOCK(B3)/BLOCK(B4)
 GRAPEVINE 426: ACT:CAUSED_BY HAND(MAX)↑ONTOP/BLOCK(B3)/
BLOCK(B4)
 GRAPEVINE 427: REAS:ACHIEVE↑STACKUP/BLOCK(B3)/BLOCK(B4)
 GRAPEVINE 428: ORD:GETONTOPOF TO FIND_SPOT↑FIND/FIND_SPOT/
INFO#39
 GRAPEVINE 429: ACT:CAUSED,BY HAND(MAX)↑RELEASE/HAND(MAX)/
BLOCK(B3)
 GRAPEVINE 430: REAS:ACHIEVE↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -75 -105 -210/TO -522 -608 -310
 GRAPEVINE 431: ACT:↑MOVE/HAND(MAX)/HAND(MAX)/FROM
-75    -105-210/TO2    -608    -310
 GRAPEVINE 432: REAS:ACHIEVE↑GRASP_STATUS[HAND(MAX)]
= GRASPING BLOCK(B6)
 GRAPEVINE 433: ACT:CAUSED,BY HAND(MAX)↑GRASP/HAND(MAX)/
BLOCK(B6)
 GRAPEVINE 434: REAS:ACHIEVE↑MOVE/HAND(MAX)/BLOCK(B6)/
FROM -522 -608 -310/TO -75 -105 -10
 GRAPEVINE 435: ACT:↑MOVE/HAND(MAX)/BLOCK(B6)/FROM
-522 -608 -310/TO -75 -105 -10
 GRAPEVINE 436: REAS:ACHIEVE↑ONTOP/BLOCK(B6)/BLOCK(B3)
 GRAPEVINE 437: ACT:CAUSED_BY HAND(MAX)↑ONTOP/BLOCK(B6)/
BLOCK(B3)
 GRAPEVINE 438: REAS:↑ASK/HUMAN(FRIEND)/HAND(MAX)
 GRAPEVINE 439: ANS:HAND(MAX) TO HUMAN(FRIEND)↑DID/
HAND(MAX)/INFO#33
```

---------- Question 52 ----------

GRAPEVINE 440: QUEST:HOW_DID/HUMAN(FRIEND) TO HAND(MAX)↑DO/
HAND(MAX)/BUILD STEEPLE( )
GRAPEVINE 441: ANS:HAND(MAX) TO HUMAN(FRIEND)↑DID/
HAND(MAX)/FROM#380/TO#439

---------- Question 53 ----------

GRAPEVINE 442: FACT:HUMAN(FRIEND) TO HAND(MAX)↑EQUAL/
PILE(GREEN|BLUE|RED)/BLOCK(COLOR = GREEN|KINDOF = -
PARALLELEPIPED|TOP_STATUS = HOLDING BLOCK(COLOR = BLUE|
KINDOF = PARALLELEPIPED|TOP_STATUS = HOLDING BLOCK(COLOR
= RED|KINDOF = PYRAMID)))

---------- Question 54 ----------

GRAPEVINE 443: QUEST:YES_NO/HUMAN(FRIEND) TO HAND(MAX)
↑EXIST/PILE(GREEN|GREEN|RED)/
GRAPEVINE 444: ANS:HAND(MAX) TO HUMAN(FRIEND)↑EXIST/
BLOCK(B4)/1

------- ------- Question 55 ----------

GRAPEVINE 445: QUEST:HOW_WOULD/HUMAN(FRIEND) TO HAND(MAX)
↑DO/HAND(MAX)/BUILD PILE(RED|GREEN|BLUE)
GRAPEVINE 446: THOUGHT:COM:HUMAN(FRIEND) TO HAND(MAX)↑BUILD/
HAND(MAX)/PILE(RED|GREEN|BLUE)
GRAPEVINE 447: THOUGHT:ORD:GETOFFOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#41
GRAPEVINE 448: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B6)/
FROM -75 -105 -10/TO -522 -608 -310
GRAPEVINE 449: THOUGHT:REAS:ACHIEVE↑ONTOP/BLOCK(B6)/
TABLE(TABL1)
GRAPEVINE 450: THOUGHT:ACT:CAUSED,BY HAND(MAX)↑OFFOF/
BLOCK(B6)/BLOCK(B3)

```
 GRAPEVINE 451: THOUGHT:REAS:ACHIEVE↑ONTOP/BLOCK(B3)/
BLOCK(B2)
 GRAPEVINE 452: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#42
GRAPEVINE 453 : THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(B6)
 GRAPEVINE 454: THOUGHT:REAS:ACHIEVE↑MOVE/HAND(MAX)/
HAND(MAX)/FROM -522 -608 -310/TO -75 -105 -210
 GRAPEVINE 455: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -522 -608 -310/TO -75 -105 -210
 GRAPEVINE 456: THOUGHT:REAS:ACHIEVE↑GRASP_STATUS[HAND(MAX)]
= GRASPING BLOCK(B3)
 GRAPEVINE 457: THOUGHT:ACT:CAUSED-BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B3)
 GRAPEVINE 458: THOUGHT:REAS:ACHIEVE↑MOVE/HAND(MAX)/
BLOCK(B3)/FROM -75 -105 -210/TO -310 -550 -160
 GRAPEVINE 459: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B3)/
FROM -75 -1.05 -210/TO -310 -550 -160
 GRAPEVINE 460: THOUGHT:REAS:ACHIEVE↑ONTOP/BLOCK(B3)/
BLOCK(B2)
 GRAPEVINE 461: THOUGHT:ACT:CAUSED,BY HAND(MAX)↑ONTOP/
BLOCK(B3)/BLOCK(B2)
 GRAPEVINE 462: THOUGHT:REAS:ACHIEVE↑STACKUP/BLOCK(B3)/
BLOCK(B2)
 GRAPEVINE 463: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#43
 GRAPEVINE 464: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(B3)
 GRAPEVINE 465: THOUGHT:REAS:ACHIEVE↑MOVE/HAND(MAX)/
HAND(MAX)/FROM -31.0 -550 -160/TO -545 145 -160
 GRAPEVINE 466: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -310 -550 -160/TO -545 145 -160
 GRAPEVINE 467: THOUGHT:REAS:ACHIEVE↑GRASP_STATUS[HAND(MAX)]
= GRASPING BLOCK(B8)
 GRAPEVINE 468: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B8)
 GRAPEVINE 469: THOUGHT:REAS:ACHIEVE↑MOVE/HAND(MAX)/
BLOCK(B8)/FROM -545 145 -160/TO -310 -550 -60
 GRAPEVINE 470: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B8)/
FROM -545 145 -160/TO -310 -550 -60
 GRAPEVINE 471: THOUGHT:REAS:ACHIEVE↑ONTOP/BLOCK(B8)/
BLOCK(B3)
```

GRAPEVINE 472: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ONTOP/
BLOCK(B8)/BLOCK(B3)
GRAPEVINE 473: THOUGHT:REAS:↑ASK/HUMAN(FRIEND)/HAND(MAX)
GRAPEVINE 474: THOUGHT:ANS:HAND(MAX) TO HUMAN(FRIEND)↑DID/
HAND(MAX)/INFO#40
GRAPEVINE 475: ANS:HAND(MAX) TO HUMAN(FRIEND)↑WOULD DO/
HAND(MAX)/FROM#445/TO#474

---------- Question 56 ----------

GRAPEVINE 476: COM:HUMAN(FRIEND) TO HAND(MAX)↑
SUBTRACT_DATABASE/HAND(MAX)/BLOCK(COLOR = GREEN|KINDOF = CUBE|
BOT_STATUS = ONTOP BLOCK(COLOR = GREEN))
GRAPEVINE 477: ORD:GETOFFOF TO FIND_SPOT↑FIND/FIND_SPOT/
INFO#44
GRAPEVINE 478: ACT:↑MOVE/HAND(MAX)/BLOCK(B6)/FROM
-75 -105 -10/TO -522 -608 -310
GRAPEVINE 479: REAS:ACHIEVE↑ONTOP/BLOCK(B6)/TABLE(TABL1)
GRAPEVINE 480: ACT:CAUSED_BY HAND(MAX)↑OFFOF/BLOCK(B6)/
BLOCK(B3)
GRAPEVINE 481: REAS:ACHIEVE↑TOP_STATUS[BLOCK(B3)] = CLEAR
GRAPEVINE 492: SUB:TMODIFY/↑SUBTRACT/HAND(MAX)/BLOCK(PNAME
= B3|HELD_STATUS = FREE|LOCASHUN = -75 -105 -210|COLOR
= GREEN|SIZE = BIG|DESCRIPTION = THE BIG GREEN CUBE]
TOP-STATUS = CLEARIEOT,STATUS = ONTOP BLOCK(B4)|KINDOF
= CUBE|DIMENSIONS = 150 150 150|XLENGTH = 150|YWIDTH = 150|
HEIGHT = 150|XCOORD = -75|YCOORD = -105|ZCOORD = -210|
LIKED-STATUS = LIKED-BY HUMAN(FRIEND)|VOLUME = 33750001
DISP_NUMB = 3|CENTER_OF_MASS = -75 -105 -285|XCMS = -75|
YCMS = -105|ZCMS = -285|SHAPE_OF_TOP = FLAT)
GRAPEVINE 483: ANS:HAND(MAX) TO HUMAN(FRIEND)↑
SUBTRACT_DATABASE/HAND(MAX)/BLOCK(B3)
GRAPEVINE 484: COM:HUMAN(FRIEND) TO HAND(MAX)↑
SUBTRACT_DATABASE/HAND(MAX)/BLOCK(COLOR = GREEN|KINDOF = CUBE)
GRAPEVINE 485: SUB:TMODIFY/↑SUBTRACT/HAND(MAX)/BLOCK(PNAME
-- B4|HELD_STATUS = FREE|LOCASHUN = -75 -105 -360|COLOR
= GREEN/SIZE = BIG/DESCRIPTION = THE BIG GREEN CUBE|
TOP_STATUS = CLEAR|BOT_STATUS = ONTOP TABLE(TABL1)|
KINDOF = CUBE|DIMENSIONS = 150 150 150|XLENGTH = 150|
YWIDTH = 150|HEIGHT = 150|XCOORD = -75|YCOORD = -1051

ZCOORD - -360|LIKED_STATUS = NOT LIKED-BY HUMAN(FRIEND)|
VOLUME 3375000|DISP_NUMB = 3|CENTER_OF_MASS = -75
-105 -435|XCMS = -75|YCMS = -105|ZCMS = -435|SHAPE_OF_TOP
= FLAT)
GRAPEVINE 486: ANS:HAND(MAX) TO HUMAN(FRIEND)↑
SUBTRACT_DATABASE/HAND(MAX)/BLOCK(B4)

---------- Question 57 ----------

GRAPEVINE 487:COM:HUMAN(FRIEND) TO HAND(MAX)↑ADD_DATABASE/
HAND(MAX)/BLOCK(COLOR = BLACK|KINDOF = CYLINDER)
GRAPEVINE 488:ORD:TMODIFY TO FIND_SPOT↑FIND/FIND_SPOT/
INFO//45
GRAPEVINE 489: ADD:TMODIFY/↑ADD/HAND(MAX)/BLOCK(BLOCK488)
GRAPEVINE 490: ANS:HAND(MAX) TO HUMAN(FRIEND)↑ADD_DATABASE/
HAND(MAX)/BLOCK(BLOCK488)
GRAPEVINE 491. : COM:HUMAN(FRIEND) TO HAND(MAX)↑ADD_DATABASE/
HAND; MAX)/BLOCK(COLOR = ORANGE|KINDOF = CYLINDER|HEIGHT
> HEIGHT[∀ BLOCK()])
GRAPEVINE 492: ORD:TMODIFY TO FIND_SPOT↑FIND/FIND_SPOT/
INFO//46
GRAPEVINE 493: ADD:TMODIFY/↑ADD/HAND(MAX)/BLOCK(BLOCK492)
GRAPEVINE 494: ANS:HAND(MAX) TO HUMAN(FRIEND)↑ADD_DATABASE/
HAND(MAX)/BLOCK(BLOCK492)

---------- Question 58 ----------

GRAPEVINE 495: COM:HUMAN(FRIEND) TO HAND(MAX)↑ADD_DATABASE/
HAND(MAX)/BLOCK(COLOR = YELLOW|KINDOF = CONE|SIZE = SMALL|
HELD_STATUS =GRASPED_BY HAND(MAX))
GRAPEVINE 496: ACT:CAUSED,BY HAND(MAX)↑RELEASE/HAND(MAX)/
BLOCK(B6)
GRAPEVINE 497: REAS:ACHIEVE↑GRASP_STATUS[HAND(MAX)] = EMPTY
GRAPEVINE 498: ACT:↑MOVE/HAND(MAX)/HAND(MAX)/FROM
-522 -608 -310/TO -522 -603 200
GRAPEVINE 499: REAS:↑ASK/HUMAN(FRIEND)/HAND(MAX)
GRAPEVINE 500:ADD:TMODIFY/↑ADD/HAND(MAX)/BLOCK(BLOCK496)
GRAPEVINE 501:ANS:HAND(MAX) TO HUMAN(FRIEND)↑ADD_DATABASE/

HAND(MAX)/BLOCK(BLOCK496)
GRAPEVINE 502: COM:HUMAN(FRIEND) TO HAND(MAX)↑ADD_DATABASE/
HAND(MAX)/BOX(DESCRIPTION = THE BOX|BOT_STATUS = ONTOP
BLOCK(COLOR = RED|SIZE = BIG))
GRAPEVINE 503: ORD:TMODIFY TO FIND_SPOT↑FIND/FIND_SPOT/
INFO//47
GRAPEVINE 504: ADD:TMODIFY/↑ADD/HAND(MAX)/BOX(BOX503)
GRAPEVINE 505: ANS:HAND(MAX) TO HUMAN(FRIEND)↑ ADD-DATABASE/

---------- Question 59 ----------

HAND(MAX)/BOX(BOX503)
GRAPEVINE 506: FACT:HUMAN(FRIEND) TO HAND(MAX)↑EQUAL/
ROCKET(PARALLELEPIPED|PYRAMID|YELLOW|BROWN|GREEN)/-
BLOCK(COLOR = YELLOW|KINDOF = PARALLELEPIPED|TOP_STATUS
= HOLDING BLOCK(COLOR = BROWN|KINDOF = PARALLELEPIPED|
TOP-STATUS = HOLDING BLOCK(COLOR = GREEN|KINDOF = PYRAMID)))

---------- Question 60 ----------

GRAPEVINE 507: COM:HUMAN(FRIEND) TO HAND(MAX)↑BUILD/
HAND(MAX)/ROCKET(CYLINDER|CONE|ORANGE|BLACK|YELLOW)
GRAPEVINE 508: ORD:GETONTOPOF TO FIND_SPOT↑FIND/FIND_SPOT/
INFO//49
GRAPEVINE 509: ORD:GETONTOPOF TO FIND_SPOT↑FIND/FIND_SPOT/
INFO//50
GRAPEVINE 510: ACT:↑MOVE/HAND(MAX)/BLOCK(BLOCK496)/
FROM -522 -608 200/TO -25 -520 -460
GRAPEVINE 511: REAS:ACHIEVE↑ONTOP/BLOCK(BLOCK496)/
TABLE(TABL1)
GRAPEVINE 512: ACT:CAUSED_BY HAND(MAX)↑ONTOP/BLOCK(BLOCK496)
/TABLE(TABL1)
GRAPEVINE 513: REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(BLOCK496)
GRAPEVINE 514: ACT:CAUSED_BY HAND(MAX)↑RELEASE/HAND(MAX)/
BLOCK(BLOCK496)
GRAPEVINE 515:REAS:ACHIEVE↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -522 -608 200/TO -570 -533 -360

GRAPEVINE 516: ACT:↑MOVE/HAND(MAX)/HAND(MAX)/FROM
-522 -608 200/TO -570 -533 -360
GRAPEVINE 51.7: REAS:ACHIEVE↑GRASP_STATUS[HAND(MAX)]
= GRASPING BLOCK(BLOCK488)
GRAPEVINE 518: ACT:CAUSED_BY HAND(MAX)↑GRASP/HAND(MAX)/
BLOCK(BLOCK488)
GRAPEVINE 519: REAS:ACHIEVE↑MOVE/HAND(MAX)/BLOCK(BLOCK488)/
FROM -570 -533 -360/TO -570 -433 -110
GRAPEVINE 520: ACT:↑MOVE/HAND(MAX)/BLOCK(BLOCK488)/
FROM . 570 -533 -360/TO -570 -433 -110
GRAPEVINE 521 :REAS:ACHIEVE↑ONTOP/BLOCK(BLOCK488)/
BLOCK(BLOCK492)
GRAPEVINE 522: ACT:CAUSED_BY HAND(MAX)↑ONTOP/BLOCK(BLOCK488)
/BLOCK(BLOCK492)
GRAPEVINE 523: REAS:ACHIEVE↑STACKUP/BLOCK(BLOCK488)/
BLOCK(BLOCK492)
GRAPEVINE 524: ORD:GETONTOPOF TO FIND_SPOT↑FIND/FIND_SPOT/
INFO#51
GRAPEVINE 525: ACT:CAUSED_BY HAND(MAX)↑RELEASE/HAND(MAX)/
BLOCK(BLOCK488)
GRAPEVINE 526: REAS:ACHIEVE↑MOVE/HAND(MAX)/HAND(MAX)/
FROM . 570 -433 -110/TO -25 -520 -460
GRAPEVINE 527: ACT:↑MOVE/HAND(MAX)/HAND(MAX)/FROM
-570 -433 -1 10/TO -25 -520 -460
GRAPEVINE 528: REAS:ACHIEVE↑GRASP_STATUS[HAND(MAX)]
= GRASPING BLOCK(BLOCK496)
GRAPEVINE 529: ACT:CAUSED_BY HAND(MAX)↑GRASP/HAND(MAX)/
BLOCK(BLOCK496)
GRAPEVINE 530: REAS:ACHIEVE↑MOVE/HAND(MAX)/BLOCK(BLOCK496)/
FROM -25 -520 -460/TO -570 -433 -60
GRAPEVINE 531: ACT:↑MOVE/HAND(MAX)/BLOCK(BLOCK496)/
FROM e-25 -520 -460/TO -570 -433 -60
GRAPEVINE 532: REAS:ACHIEVE↑ONTOP/BLOCK(BLOCK496)/
BLOCK(BLOCK488)
GRAPEVINE 533: ACT:CAUSED_BY HAND(MAX)↑ONTOP/BLOCK(BLOCK496)
/BLOCK(BLOCK488)
GRAPEVINE 534: REAS:↑ASK/HUMAN(FRIEND)/HAND(MAX)
GRAPEVINE 535: ANS:HAND(MAX) TO HUMAN(FRIEND)↑DID/
HAND(MAX)/INFO#48

---------- Question 61 ----------

GRAPEVINE 536: COM:HUMAN(FRIEND) TO HAND(MAX)↑FIND/
HAND(MAX)/OBJECT(BOT_STATUS = ONTOP TABLE(DEFINITE))
GRAPEVINE 537: ANS:HAND(MAX) TO HUMAN(FRIEND)↑FIND/
HAND(MAX)/BLOCK(VAR B2 B7 B5 B6 BLOCK492)
GRAPEVINE 538: ANS:HAND(MAX) TO HUMAN(FRIEND)↑ FIND/
HAND(MAX)/BOX(VAR BOX1)

---------- Question 62 ----------

GRAPEVINE 539: FACT:HUMAN(FRIEND) TO HAND(MAX)↑EQUAL/
GNAME[BLOCK(B2)]/SUPERBLOCK

---------- Question 63 ----------

GRAPEVINE 540: QUEST:WHEN_DID/HUMAN(FRIEND) TO HAND(MAX)↑DO/
HAND(MAX)/PICKUP BLOCK(GNAME = SUPERBLOCK)
GRAPEVINE 541. : ACT:NOAP
GRAPEVINE 542: REAS:FIND_NUMB/↑ NOT EXIST/RELATION/
GRAPEVINE 543: ACT:NOAP
GRAPEVINE 544: REAS:FIND_NUMB/↑NOT EXIST/RELATION/
GRAPEVINE 545: ANS:HAND(MAX) TO HUMAN(FRIEND)↑DID/
HAND(MAX)/#15/WHILE#19

---------- Question 64 ----------

GRAPEVINE 546:THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(BLOCK496)/
FROM -570 -433 -60/TO -25 -520 -460
GRAPEVINE 547: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 548: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#52
GRAPEVINE 549: THOUGHT:ACT:NOAP
GRAPEVINE 550: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(BLOCK496)/FROM -25 -520 -460/TO -25 -520 -460
GRAPEVINE 551: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ONTOP/

BLOCK(BLOCK496)/TABLE(TABL1)
GRAPEVINE 552: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(BLOCK496)
GRAPEVINE 553: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(BLOCK496)
GRAPEVINE 554: THOUGHT:REAS:1 ASK/AT-TIME/RELEASE
GRAPEVINE 555: THOUGHT:ACT:1 MOVE/HAND(MAX)/HAND(MAX)/
FROM -25 -520 -460/TO -570 -433 -110
GRAPEVINE 556: THOUGHT:REAS:1 ASK/AT_TIME/MOVETO
GRAPEVINE 557: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ GRASP/
HAND(MAX)/BLOCK(BLOCK488)
GRAPEVINE 558: THOUGHT:REAS:1 ASK/AT-TIME/GRASP
GRAPEVINE 559: THOUGHT:ACT:1 MOVE/HAND(MAX)/BLOCK(BLOCK488)/
FROM -570 -433 -110/TO -570 -533 -360
GRAPEVINE 560: THOUGHT:REAS:1 ASK/AT_TIME/MOVETO
GRAPEVINE 561 : THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND SPOT/INFO#53
GRAPEVINE 562: THOUGHT:ACT:NOAP
GRAPEVINE 563: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(BLOCK488)/FROM -570 -533 -360/TO -570 -533 -360
GRAPEVINE 564: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ONTOP/
BLOCK(BLOCK488)/TABLE(TABL1)
GRAPEVINE 565: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(BLOCK488)
GRAPEVINE 566: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(BLOCK488)
GRAPEVINE 567: THOUGHT:REAS:1 ASK/AT_TIME/RELEASE
GRAPEVINE 568: THOUGHT:ACT:1 MOVE/HAND(MAX)/HAND(MAX)/
FROM -570 -503 -360/TO -522 -608 200
GRAPEVINE 569: THOUGHT:REAS:1 ASK/AT_TIME/MOVETO
GRAPEVINE 570: THOUGHT:ACT:1 MOVE/HAND(MAX)/HAND(MAX)/
FROM -522 -608 200/TO - 25 -520 -460
GRAPEVINE 571: THOUGHT:REAS:ACHIEVE↑GRASP/HAND(MAX)/
BLOCK(BLOCK496)
GRAPEVINE 572: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(BLOCK496)
GRAPEVINE 573: THOUGHT:REAS:1 ASK/AT_TIME/GRASP
GRAPEVINE 574: THOUGHT:ACT:1 MOVE/HAND(MAX)/BLOCK(BLOCK496)/
FROM - 25 -520 -460/TO -522 -608 200
GRAPEVINE 575: THOUGHT:REAS:1 ASK/AT_TIME/MOVETO
GRAPEVINE 576: THOUGHT:SUB:TMODIFY/↑SUBTRACT/HAND(MAX)/
BOX(PNAME = BOX503|HELD_STATUS - - FREE|LOCASHUN = -310

-550 -160|SIZE = MEDIUM-SIZED|DESCRIPTION = THE BOX|
TOP_STATUS = CLEAR|BOT_STATUS = ONTOP BLOCK(B2)|DIMENSIONS
-- 100 100 150|XLENGTH = 100|YWIDTH = 100|HEIGHT = 150|
XCOORD = -310|YCOORD = -550|ZCOORD = -160|VOLUME = 15000001
DISP_NUMB = 2|CENTER_OF_MASS = -310 -550 -235|XCMS = -310|
YCMS -- -550|ZCMS = -235|SHAPE_OF_TOP = FLAT)
GRAPEVINE 577: THOUGHT:SUB:TMODIFY/↑SUBTRACT/HAND(MAX)/
BLOCK(PNAME = BLOCK496|HELD_STATUS = GRASPED-BY HAND(MAX)|
LOCASHUN = -522 -608 200|COLOR = YELLOW|SIZE = SMALL|
DESCRIPTION = THE SMALL YELLOW CONE|TOP_STATUS = CLEAR|
BOT_STATUS = FREE|KINDOF = CONE|DIMENSIONS = 50 75 50|
XLENGTH = 50|YWIDTH -- 75|HEIGHT = 50|XCOORD = -522|
YCOORD = -608|ZCOORD = 200|VOLUME = 73623|DISP_NUMB = 2|
CENTER_OF_MASS = -522 -608 175|XCMS = -522|YCMS = -608|
ZCMS = 175|SHAPE_OF_TOP = POINTED)
GRAPEVINE 578: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -522 -608 200/TO -522 -608 -310
GRAPEVINE 579: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 580: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B6)
GRAPEVINE 581: THOUGHT:REAS:↑ASK/AT_TIME/GRASP
GRAPEVINE 582: THOUGHT:SUB:TMODIFY/↑SUBTRACT/HAND(MAX)/
BLOCK(PNAME - BLOCK492|HELD_STATUS = FREE|LOCASHUN
= -570 -433 -260)COLOR = ORANGE|SIZE = MEDIUM-SIZED]
DESCRIPTION = THE MEDIUM-SIZED ORANGE CYLINDER|TOP_STATUS
= CLEAR|BOT_STATUS = ONTOP TABLE(TABL1)|KINDOF = CYLINDER|
DIMENSIONS = 100 100 250|XLENGTH = 100|YWIDTH = 100|
HEIGHT = 250|XCOORD = -570|YCOORD = -433|ZCOORD = -260|
VOLUME = 1963493|DISP_NUMB = 2|CENTER_OF_MASS = -570
-433 -385|XCMS = -570|YCMS = -433|ZCMS = -385|SHAPE_OF_TOP
- FLAT)
GRAPEVINE 583: THOUGHT:SUB:TMODIFY/↑SUBTRACT/HAND(MAX)/
BLOCK(PNAME = BLOCK488|HELD_STATUS = FREE|LOCASHUN
= -570 -533 -360|COLOR = BLACK/SIZE = MEDIUM-SIZED|
DESCRIPTION = THE MEDIUM-SIZED BLACK CYLINDER|TOP_STATUS
= CLEAR|BOT_STATUS = ONTOP TABLE(TABL1)|KINDOF = CYLINDER]
DIMENSIONS = 100 I. 00 150|XLENGTH = 100|YWIDTH = 100|
HEIGHT = 150|XCOORD = -570|YCOORD = -533|ZCOORD = -360|
VOLUME = 1178096|DISP_NUMB = 3|CENTER_OF_MASS = -570
-533 -435|XCMS = -570|YCMS = -533|ZCMS = -435|SHAPE_OF_TOP
- FLAT)
--GRAPEVINE 584: THOUGHT:ORD:TMODIFY TO FIND_SPOT↑FIND/

```
FIND_SPOT/INFO#54
 GRAPEVINE 585: THOUGHT:ADD:TMODIFY/↑ADD/HAND(MAX)/BLOCK(B4)
 GRAPEVINE 586: THOUGHT:ORD:TMODIFY TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#55
 GRAPEVINE 587: THOUGHT:ADD:TMODIFY/↑ADD/HAND(MAX)/BLOCK(B3)
 GRAPEVINE 588: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B6)/
FROM -522 -608 -310/TO -75 -105 -10
 GRAPEVINE 589: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
 GRAPEVINE 590: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B6)/
FROM -75 -105 -10/TO -522 -603 -310
 GRAPEVINE 591: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
 GRAPEVINE 592: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#56
 GRAPEVINE 593: THOUGHT:ACT:NOAP
 GRAPEVINE 594: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B6)/FROM -522 -608 -310/TO -522 -608 -330
 GRAPEVINE 595: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ONTOP/
BLOCK(B6)/TABLE(TABL1)
 GRAPEVINE 596: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B6)
 GRAPEVINE 597: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(B6)
 GRAPEVINE 598: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
 GRAPEVINE 599: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -522 -608 -310/TO -75 -105 -210
 GRAPEVINE 600: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
 GRAPEVINE 601: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B3)
 GRAPEVINE 602: THOUGHT:REAS:↑ASK/AT_TIME/GRASP
 GRAPEVINE 603: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B3)/
FROM -75 -105 -210/TO -310 -550 -360
 GRAPEVINE 604: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
 GRAPEVINE 605: THOUGHT:ORD:GETONTOPOF TO FIND-SPOT? FIND/
FIND_SPOT/INFO#57
 GRAPEVINE 606: THOUGHT:ACT:NOAP
 GRAPEVINE 607: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B3)/FROM -310 -550 -160/TO -310 -550 -160
 GRAPEVINE 608: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ONTOP/
BLOCK(B3)/BLOCK(B2)
 GRAPEVINE 609: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B3)
 GRAPEVINE 610: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
```

HAND(MAX)/BLOCK(B3)
GRAPEVINE 611: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
GRAPEVINE 612: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -310 -550 -160/TO -522 -608 -310
GRAPEVINE 613: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 614: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B6)
GRAPEVINE 615: THOUGHT:REAS:↑ASK/AT_TIME/GRASP
GRAPEVINE 616: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B6)/
FROM e-522 -608 -310/TO -105 -70 -160
GRAPEVINE 617: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 618: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#58
GRAPEVINE 619: THOUGHT:ACT:NOAP
GRAPEVINE 620: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B6)/FROM -105 -70 -160/TO -105 -70 -160
GRAPEVINE 621: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ONTOP/
BLOCK(B6)/BLOCK(B4)
GRAPEVINE 622: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B6)
GRAPEVINE 623: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(B6)
GRAPEVINE 624: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
GRAPEVINE 625: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -105 -70 -160/TO -472 -608 -460
GRAPEVINE 626: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 627: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B5)
GRAPEVINE 628: THOUGHT:REAS:↑ASK/AT_TIME/GRASP
GRAPEVINE 629: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B5)/
FROM -472 -608 -460/TO -55 -155 -310
GRAPEVINE 630: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 631: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#59
GRAPEVINE 632: THOUGHT:ACT:NOAP
GRAPEVINE 633: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B5)/FROM -55 -155 -310/TO -55 -155 -310
GRAPEVINE 634: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ONTOP/
BLOCK(B5)/BLOCK(B4)
GRAPEVINE 635: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B5)
GRAPEVINE 636: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/

tIAND(MAX)/BLOCK(B5)
GRAPEVINE 637: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
GRAPEVINE 638: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -55 -155 -310/TO -422 -608 -460
GRAPEVINE 639: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 640: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑GRASP/
tIAND(MAX)/BLOCK(B7)
GRAPEVINE 641: THOUGHT:REAS:↑ASK/AT_TIME/GRASP
GRAPEVINE 642: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B7)/
FROM -422 -608 -460/TO -55 -155 -260
GRAPEVINE 643: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 644: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#60
GRAPEVINE 645: THOUGHT:ACT:NOAP
GRAPEVINE 646: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B7)/FROM -55 -155 -260/I-O -55 -155 -260
GRAPEVINE 647: THOUGHT:ACT:CAUSED,BY HAND(MAX)↑ONTOP/
BLOCK(B7)/BLOCK(B5)
GRAPEVINE 648: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B7)
GRAPEVINE 649: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(B7)
GRAPEVINE 650: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
GRAPEVINE 651: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -55 -155 -260; TO -545 145 -160
GRAPEVINE 652: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 653: THOUGHT:ACT:CAUSED,BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B8)
GRAPEVINE 654: THOUGHT:REAS:↑ASK/AT_TIME/GRASP
GRAPEVINE 655: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B8)/
FROM -545 145 -160/TO -470 165 -410
GRAPEVINE 656: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 657: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#61
GRAPEVINE 653: THOUGHT:ACT:NOAP
GRAPEVINE 659: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B8)/FROM -470 165 -410/TO -470 165 -410
GRAPEVINE 660: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ONTOP/
BLOCK(B8)/TABLE(TABL1)
GRAPEVINE 661: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B8)
GRAPEVINE 662: THOUGHT:ACT:CAUSED,BY HAND(MAX)↑RELEASE/

HAND(MAX)/BLOCK(B8)
GRAPEVINE 663: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
GRAPEVINE 664: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -470 I. 65 -410/TO -55 -155 -260
GRAPEVINE 665: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 666: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B7)
GRAPEVINE 667: THOUGHT:REAS:↑ASK/AT_TIME/GRASP
GRAPEVINE 668: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B7)/
FROM -55 -155 -260/TO -422 -608 -460
GRAPEVINE 66 9: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 6 70: THOUGH-I-:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
f IND_SPOT/INFO#62
GRAPEVINE 6 71: THOUGHT:ACT:NOAP
GRAPEVINE 6 72: THOUGH T:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B7)/FROM -422 -608 -460/TO -422 -608 -460
GRAPEVINE 6 73: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ONTOP/
BLOCK(B7)/TABLE(TABL1)
GRAPEVINE 6 74: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B7)
GRAPEVINE 6 75: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
tIAND(MAX)/BLOCK(B7)
GRAPEVINE 6 76: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
GRAPEVINE 677: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -422 -608 -460/TO -55 -155 -310
GRAPEVINE 6 78: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 6 79: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B5)
GRAPEVINE 680: THOUGHT:REAS:↑ASK/AT_TIME/GRASP
GRAPEVINE 681: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B5)/
FROM -55 -155 -310/TO -310 -550 -110
GRAPEVINE 682: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 683: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B5)/
FROM -310 -550 -110/TO -75 -450 -460
GRAPEVINE 684: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 685: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#63
GRAPEVINE 686: THOUGHT:ACT:NOAP
GRAPEVINE 657: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B5)/FROM -75 -450 -460/TO -75 -450 -460
GRAPEVINE 688: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ONTOP/
BLOCK(B5)/TABLE(TABL1)

GRAPEVINE 689: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B5)
GRAPEVINE 690: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(B5)
GRAPEVINE 691: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
GRAPEVINE 692: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -75 -450 -460/TO -422 -608 -460
GRAPEVINE 6 93: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 694: THOUGHT:ACT:CAUSED,BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B7)
GRAPEVINE 695: THOUGHT:REAS:↑ASK/AT_TIME/GRASP
GRAPEVINE 696: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B7)/
FROM -422 -608 -460/TO -75 -450 -410
GRAPEVINE 697: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 698: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#64
GRAPEVINE 699: THOUGHT:ACT:NOAP
GRAPEVINE 700: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B7)/FROM -75 -450 -410/TO -75 -450 - 4 1 0
GRAPEVINE 701. : THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ONTOP/
BLOCK(B7)/BLOCK(B5)
GRAPEVINE 702: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B7)
GRAPEVINE 703: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(B7)
GRAPEVINE 704: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
GRAPEVINE 705: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -75 -450 -410/TO -310 -550 -160
GRAPEVINE 706: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 70 7: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B3)
GRAPEVINE 788: THOUGHT:REAS:↑ ASK/AT-TIME/GRASP
GRAPEVINE 70 9: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B3)/
FROM -310 -550 -160/TO -545 -255 -360
GRAPEVINE 710: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 7.11: THOUGHT:ORD:GETONTOPOF TO FIND,SPOTT FIND/
FIND_SPOT/INFO#65
GRAPEVINE 71.2: THOUGHT:ACT:NOAP
GRAPEVINE 713: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B3)/FROM -545 -255 -360/TO -545 -255 -360
GRAPEVINE 714: THOUGHT:ACT:CAUSED,BY HAND(MAX)↑ONTOP/
BLOCK(B3)/TABLE(TABL1)

GRAPEVINE 715: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B3)
GRAPEVINE 71.6: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(B3)
GRAPEVINE 717: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
GRAPEVINE 718: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -545 -255 -360/TO -545 145 -260
GRAPEVINE 719: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 720: THOUGHT:ACT:CAUSED,BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B1)
GRAPEVINE 721: THOUGH-I":REAS:I'ASK/AT-TIME/GRASP
GRAPEVINE 722: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B1)/
FROM -545 145 -260/TO -475 -105 -260
GRAPEVINE 723: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 724: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑ FIND/
FIND_SPOT/INFO#66
GRAPEVINE 725: THOUGH-J:ACT:NOAP
GRAPEVINE 726: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B1)/FROM -475 -105 -260/TO -475 -105 -260
GRAPEVINE 72 7: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑ONTOP/
BLOCK(B1)/TABLE(TABL1)
GRAPEVINE 725: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B1)
GRAPEVINE 729: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(B1)
GRAPEVINE 730: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
GRAPEVINE 731: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -475 -105 -260/TO -310 -550 200
GRAPEVINE 732: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO
GRAPEVINE 733: THOUGHT:ACT:↑MOVE/HAND(MAX)/HAND(MAX)/
FROM -33 0 -550 200/TO -310 -550 -310
GRAPEVINE 734: THOUGHT:REAS:ACHIEVE↑GRASP/HAND(MAX)/
BLOCK(B2)
GRAPEVINE 735: THOUGHT:ACT:CAUSED_BY HAND(MAX)↑GRASP/
HAND(MAX)/BLOCK(B2)
GRAPEVINE 736: THOUGHT:REAS:1 ASK/AT-TIME/GRASP
GRAPEVINE 737: THOUGHT:ACT:↑MOVE/HAND(MAX)/BLOCK(B2)/
FROM -31.0 -550 -310/TO -310 -550 200
GRAPEVINE 738: THOUGH-i-:I?EAS:TASK/AT-TIME/MOVETO
GRAPEVINE 739: THOUGHT:ACT: ↑MOVE/HAND(MAX)/BLOCK(B2)/
FROM -310 -550 200/TO -310 -550 -310
GRAPEVINE 740: THOUGHT:REAS:↑ASK/AT_TIME/MOVETO

GRAPEVINE 741: THOUGHT:ORD:GETONTOPOF TO FIND_SPOT↑FIND/
FIND_SPOT/INFO#67
GRAPEVINE 742: THOUGHT:ACT:NOAP
GRAPEVINE 743: THOUGHT:REAS:ALREADY_DONE↑MOVE/HAND(MAX)/
BLOCK(B2)/FROM -310 -550 -310/TO -310 -550 -310
GRAPEVINE 744: THOUGHT:ACT:CAUSED,BY HAND(MAX)↑ONTOP/
BLOCK(B2)/TABLE(TABL1)
GRAPEVINE 745: THOUGHT:REAS:ACHIEVE↑RELEASE/HAND(MAX)/
BLOCK(B2)
GRAPEVINE 746: THOUGHT:ACT:CAUSED,BY HAND(MAX)↑RELEASE/
HAND(MAX)/BLOCK(B2)
GRAPEVINE 747: THOUGHT:REAS:↑ASK/AT_TIME/RELEASE
GRAPEVINE 748: QUEST:HOW_MANY/HUMAN(FRIEND) TO HAND(MAX)
↑EXIST/BLOCK(KINDOF ≠ PYRAMID|LOCASHUN LEFT-OF BOX())/
GRAPEVINE 749: ANS:HAND(MAX) TO HUMAN(FRIEND)↑EXIST/
BLOCK(VAR B1 B2 B5 B4 B3)/5

---------- Question 65 ----------

GRAPEVINE 750: QUEST:WHY_DID/HUMAN(FRIEND) TO HAND(MAX)↑DO/
↑HAND(MAX)/DROP BLOCK(GNAME = SUPERBLOCK)
GRAPEVINE 751: ACT:NOAP
GRAPEVINE 752: REAS:MATCHX/↑NOT EXIST/STRAQ/ACT:↑DROP/
HAND(MAX)/BLOCK(GNAME = SUPERBLOCK) .
GRAPEVINE 753: ANS:HAND(MAX) TO HUMAN(FRIEND)↑NOT REMEMBER/
HAND(MAX)/INFO#68

---------- Question 66 ----------

GRAPEVINE 754: QUEST:YES_NO/HUMAN(FRIEND) TO HAND(MAX)
↑EXIST/OBJECT(LOCASHUN IN_FRONT_OF BLOCK(COLOR = RED|
KINDOF = PYRAMID))/
GRAPEVINE 755: ANS:HAND(MAX) TO HUMAN(FRIEND)↑EXIST/
BLOCK(VAR B2 E37 B5)/4
GRAPEVINE 756: ANS:HAND(MAX) TO HUMAN(FRIEND)↑EXIST/
BOX(VAR BOX503)/4

```
INFO 1: LOCATION ONTOPOF TABLE(TABL1)
INFO 2: COM:HUMAN( FRIEND) TO HAND(MAX)↑PICKUP/HAND(MAX)/BLOCK(B2)
INFO 3: COM:HUMAN(FRIEND) TO HAND(MAX)↑GRASP/HAND(MAX)
      /BLOCK(DEFINITE|KINDOF = PYRAMID)
INFO 4: LOCATION ONTOPOF BOX(BOX1)
INFO 5: LOCATION ONTOPOF TABLE(TABL1)
INFO 6: COM:HUMAN( FRIEND) TO HAND(MAX)↑INSIDE/BLOCK(B1)/BOX(BOX1)
INFO 7: ONTOP/BLOCK(B6)/BLOCK(B4)
INFO 8: LOCATION ONTOPOF BLOCK(B8)
INFO 9: I-t IOUGHT:COM:HUMAN(FRIEND) TO HAND(MAX)↑ONTOP
      /BLOCK(KINDOF = PYRAMID)/BLOCK(KINDOF = PYRAMID)
INFO 1 0: LOCATION ONTOPOF TABLE( TABL1)
INFO 11: LOCATION ONTOPOF BLOCK(B2)
INFO I 2: LOCATION ONTOPOF BLOCK(B5)
INFO 13: THOUGHT:COM:HUMAN(FRIEND) TO HAND(MAX)↑STACKUP
      /HAND(MAX)/BLOCK(B2) BLOCK(B5) AND BLOCK(B6)
INFO 3 4: LOCATION ONTOPOF BLOCK(B2)
INFO 1 5: LOCATION ONTOPOF TABLE(TABL1)
INFO 16: LOCATION ONTOPOF BLOCK(B3)
INFO 1.7: COM:HUMAN(FRIEND) TO HAND(MAX)↑STACKUP/HAND(MAX)
      /BLOCK(B2) BLOCK(B3) AND BLOCK(B5)
INFO 1 8:LOCATION ONTOPOF BLOCK(B4)
INFO 1 9: COM:HUMAN( FRIEND) TO HAND(MAX)↑ONTOP/BLOCK(B5)/BLOCK( B4)
INFO 20: LOCATION ONTOPOF BLOCK(B5)
INFO 21: COM:HUMAN(FRIEND) TO HAND(MAX)↑ONTOP/BLOCK(B7)/BLOCK(B5)
INFO22: LOCATION ONTOPOF TABLE(TABL1)
INFO 23: LOCATION ONTOPOF TABLE(TABL1)
INFO 24: LOCATION ONTOPOF BLOCK(B5)
INFO 35: LOCATION ONTOPOF TABLE(TABL1)
INFO 26: LOCATION ONTOPOF TABLE(TABL1)
INFO 27: LOCATION ONTOPOF BLOCK(B5)
INFO 23: LOCATION ONTOPOF TABLE(TABL1)
INFO 29: LOCATION ONTOPOF TABLE(TABL1)
INFO 30: LOCATION ONTOPOF BLOCK(B5)
INFO 31: LOCATION ONTOPOF BLOCK(B1)
INFO 32: COM:HUMAN(FRIEND) TO HAND(MAX)↑ONTOP/BLOCK(B8)/BLOCK(B1)
INFO 33: COM:HUMAN(FRIEND) TO HAND(MAX)↑BUILD/HAND(MAX)/STEEPLE()
INFO 34: LOCATION ONTOPOF CLOSK)B49
INFO 35: LOCATION ONTOPOF BLOCK(B4)
```

```
INFO 36: LOCATION ONTOPOF TABLE(TABL1)
INFO 37: LOCATION ONTOPOF TABLE(TABL1)
INFO 38: LOCATION ONTOPOF TABLE(TABL1)
INFO 39: LOCATION ONTOPOF BLOCK(B3)
INFO 40: THOUGHT:COM:HUMAN(FRIEND) TO HAND(MAX)↑BUILD
     /HAND(MAX)/PILE(RED|GREEN|BLUE)
INFO 41: LOCATION ONTOPOF TABLE(TABL1)
INFO 42: LOCATION ONTOPOF BLOCK(B2)
INFO 43: LOCATION ONTOPOF BLOCK(B3)
INFO 44: LOCATION ONTOPOF TABLE(TABL1)
INFO 45: LOCATION ONTOPOF TABLE(TABL1)
INFO 46: LOCATION ONTOPOF TABLE(TABL1)
INFO 47: LOCATION ONTOPOF BLOCK(B2)
INFO 48:COM:HUMAN(FRIEND) TO HAND(MAX)↑BUILD/HAND( MAX)
     /ROCKET(CYLINDER|CONE|ORANGE|BLACK|YELLOW)
INFO 49: LOCATION ONTOPOF BLOCK(BLOCK492)
INFO 50: LOCATION ONTOPOF TABLE(TABL1)
INFO 51: LOCATION ONTOPOF BLOCK(BLOCK488)
INFO 52: LOCATION ONTOPOF TABLE(TABL1)
INFO 53: LOCATION ONTOPOF TABLE(TABL1)
INFO 54: LOCATION ONTOPOF TABLE(TABL1)
INFO 55: LOCATION ONTOPOF BLOCK(B4)
INFO 56: LOCATION ONTOPOF TABLE(TABL1)
INFO 57: LOCATION ONTOPOF BLOCK(B2)
INFO 58: LOCATION ONTOPOF BLOCK(B4)
INFO 59: LOCATION ONTOPOF BLOCK(B4)
INFO 60: LOCATION ONTOPOF BLOCK(B5)
INFO 61 : LOCATION ONTOPOF TABLE(TABL1)
INFO 62: LOCATION ONTOPOF TABLE(TABL1)
INFO 63: LOCATION ONTOPOF TABLE(TABL1)
INFO 64: LOCATION ONTOPOF BLOCK(B5)
INFO 65: LOCATION ONTOPOF TABLE(TABL1)
INFO 66: LOCATION ONTOPOF TABLE(TABL1)
INFO 67: LOCATION ONTOPOF TABLE(TABL1)
INFO 68: ACT:↑DROP/HAND(MAX)/BLOCK(GNAME = SUPERBLOCK)
```

REFERENCES

1. J. McCarthy, "Programs with Common Sense", Mechanization of Thought Processes, Vol. 1, London: HMSO (1959).

2. A. Newell, J. C. Shaw, and H. A. Simmon, "Elements of a Theory of Human Problem Solving", Psychological Review 65, p151 (1958).

3. S. Amarel, "On the Representation Problerns of Reasoning about Action", in D. Mitchie(Ed.) Machine Intelligence 3, New York: American, Elsevier (1968).

4. J. McCarthy and P.J. Hayes, "Some Philosophical Problems from the Standpoint of Artificial Intelligence", in B. Meltzer and D. Michie (Eds.) Machine Intelligence 4, p463, Edinburg (1969).

5. M.R. Quillian, "Semantic Memory", in M. Minsky (Ed.), Semantic Information Processing, Cambridge, Mass. M.I.T. Press, 1968, "The Teachable Language Comprehender", Communications of the ACM 12, p459 (1969).

6. T. Winograd, "Procedures as a Representation for Understanding Natural Language", MAC TR-84 M. I. T. Artificial Intelligence Laboratory, Ph. D. thesis (1971).
"Understanding Natural Language", New York, Academic Press (1972).

7. R. C. Shank, "Conceptual Dependency: A Theory of NaturalLanguage Understanding", Cogn.Psychol., Vol. 3, 552 (1972).

8. P. M. Winston, "Learning Structural Descriptions form Examples", MAC TR-76, M. I. T. Artificial Intelligence Laboratory Report, Ph. D. thesis (1970).

9. T. O. Binford, "Visual Perception by Computer" in Systems, Science and Cybernetics, Miami. (Dec.1971).

10. G. J. Agin and T. O. Binford, "Computer Description of Curved Objects", p629, in Proceedings Third International Joint Conference on Artificial Intelligence, Stanford Research Institute, Menlo Park (Stanford, Aug. 1973).

11. C. Hewitt, "PLANNER: A Language for Manipulating Models and Proving Theorems in a Robot", M. I. T. Artificial Intelligence Laboratory Memo 168 (Aug. 1970).

12. J. F. Rulifson, J. A. Derksen, and R. J. Waldinger, "QA4: A Procedural Calculus for Intuitive Reasoning", Tech. Note 74, Artificial Intelligence Center SRI (Nov.1972).

13. D. V. McDermott and G. J. Sussman, "The Conniver Reference Manual", M. I. T. Artificial Intelligence Laboratory Memo 259 (May,1972).

14. C. Hewitt, P. Bishop, and R. Steiger, "A Universal Modular Act or Formalism for Artificial Intelligence", p235, in Proceedings Third International Joint Conference on Artificial Intelligence, Stanford Research Institute, Menlo Park (Stanford, Aug. 1973).

15. E. A. Feigenbaum, "A Simulation in Verbal Learning Behavior", in E. A. Feigenbaum and Feldman (Ed.), Computers and Thought. McGraw-Hill, New York, (1963).
H. A. Simon and E. A. Feigenbaum, "An Information Processing Theory of Some Effects of Similarity, Familiarity, and Meaningfulness in Verbal Learning", Journal of Verbal Learning and Verbal Behavior, Vol. 3, p385(1964).

16. D. L. Hintzman, "Explorations with a Discrimination Net Model for Paired-associate Learning", Journal of Mathematical Psychology, Vol. 5, p123. (1968).

17. D. E. Rurnelhart, P. H. Lindsay, and D. A. Norman, "A Process Model for Long-term Memory", in E. Tulving and W. Donaldson (Eds.), Organization and Memory. Academic Press, New York (1972).

18. J. R. Anderson and G. Bower, Human Associative Memory. V. t-i. Winston and Sons, Washington, D. C. (1973).

19. See SAIL MANUAL, K. A. VanLehn, Editor, Stanford Artificial Intelligence Project, Memo No. 204 (June 1973).

20. T. Winograd, "A Procedural Model of Language Understanding", in R. C. Shank and K. M. Colby (Eds.), Computer Models of Thought and Language, W. H. Freeman and Company, San Francisco, p152 (1973).

21. D. C. Smith and H. J. Enca, "Backtracking in MLISP2", p6 77, in Proceedings Third International Joint Conference on Artificial Intelligence, Stanford Research Institute, Menlo Park (Stanford, Aug. 1973).

22. T. Winograd, "Breaking the Complexity Barrier Again", in Proceedings of the ACM, SIGPLAN-SIGIR Interface Meeting, (Nov. 1973) (to be published).

23. N. M. Goldman and C. K. Riesbeck, "A Conceptually Based Sentence Paraphraser", Stanford Artificial Intelligence Laboratory Memo No. 196 (May 1973).

24. R. C. Shank and C. J. Rieger III, "Inference and Computer Understanding of Natural Language", AIM 197 (May, 1973).

25. C. J. Rieger III, "Conceptual Memory", Stanford Artificial Intelligence Laboratory, Ph. D. thesis (1974).

26. G. J. Sussman, "A Computational Model of Skill Acquisition", M. I. T. Artificial Intelligence Laboratory, Ph. D. thesis (Aug. 1973).