Name:

Examination # **2** Date: **11/17/98**

Please indicate your section, or if you do not remember, put a tick mark on the last line.

Time	Instructor	
10-11	Davis	
11-12	Davis	
10-11	Szolovits	
11-12	Szolovits	
1-2	Shrobe	
1-2	Berwick	
2-3	Berwick	
2-3	Lozano-Perez	
3-4	Lozano-Perez	

Please indicate the TA you are assigned to.

Calista Tait	
Dennis Ruhl	
Irene Wilson	
Jonathan Loflin	
Ozlem Uzuner	
Serkan Arslanalp	

Problem 1	
Problem 2	
Problem 3	
Problem 4	
Total	

Problem 1 Framing the question (26 points)

True to their name, MicroMonopoly (MM) wants to expand their business from software to the hardware business so they can "control everything right down to the metal".

Ben Bitdidle has been hired by MM to see if they can add a natural language interface to MM's system that automatically assembles computer components via language commands and questions.

MM has already assembled a relational database to describe their domain, which looks very similar to the "tools" database on page 604 of the Winston textbook or the "books" database of Problem set 7, along with the relational database operators in that problem set. (We have provided all examples of the relational operators that you will need in the extracted code on the next page.)

ID	CLASS	COLOR	SIZE	WEIGHT (OZ)	LOCATION
SI1	Soldering-iron	black	medium	10	Toolbox
CB-101	Circuit-board	green	small	3	Workbench
CB-333	Circuit-board	green	medium	3	Toolbox
TB1	Trackball	white	small	4	Toolbox
M3	Mouse	gray	small	3.5	Toolbox
T1	Toolbox	silver	medium	40	Workbench
T2	Toolbox	silver	large	40	Floor
P1	Pegboard	white	large	30	Wall
W1	Workbench	brown	large	32	Assembly-room

Naturally, Ben starts hacking with something he already knows: the semantic transition tree grammar from Problem set 7, which in typical fashion MicroMonopoly has already obtained by buying out a company, in this case, MIT, and modifying the code for its own ends.

Note that the code for Problem set 7's original grammar, without the changes asked for in the problems, has been augmented to include network routines for place phrases. There are no other changes, other than the network routines for places phrases. The entire original grammar from Problem set 7, with the place-phrase code added, is provided at the end of this examination. The place-phrase code is located at the end of the grammar. You can tear off the grammar for ease in manipulation, if you like. You need not hand the grammar pages in.

Part A (6 points)

MM has written the low-level routine for a robot arm, `put-on`, which takes two objects `o1` and `o2` and has the robot arm move `o1` to the location specified by `o2`.

MM hands Ben his first assignment, which is to connect appropriate English commands to the activation of the `put-on` routine. The following is a representative example of such an appropriate command:

Put the trackball on the workbench

Ben comes up with a possible change to the existing semantic transition network routine `main`, and sets aside this work for an espresso break. When he comes back, he finds that his pet poikilotherm has chewed up part of his code. Rather than reconstruct his code, Ben asks you to help by filling in the missing part below. Only one or two lines are needed, depending on how you lay out your work.

```
;;;; MAIN TREE
(define main
  (make-tree
   '(branch
     ;; COUNT THE CIRCUIT-BOARDS
     (count (object-description --> o)
           if-end-rtn (db-call `(db-report-count (db-count-items ,o))))
     (what are (object-description --> o) (attribute-list --> a-l)
           if-end-rtn (db-call `(db-show (db-project ,o '(over ,@a-l)))))

;;; ADDED BY BEN
     (







          `(put-on ,o1 ,o2 ))
;;;;
;;;; END OF BEN'S ADDITIONS
     ((identify)
      branch
      ;; IDENTIFY THE CIRCUIT-BOARDS
      ((object-description --> o)
      if-end-rtn (db-call `(db-show ,o)))
     (stop if-end-rtn 'quit)
     (rtn (format #t "~%Sorry, I do not understand")
          #f))))
```

Part B (6 points)

In order to have the transition network successfully parse the new command, does Ben have to make any *other* changes besides the changes to `main`? Check the YES or NO box and then in a few words explain Why or Why Not:

☐ Yes, Ben must make other changes, because:

☐ No, Ben need not make other changes, because:

Part C (6 points)

Ben is happy to get an answer from you that works — until another memo arrives that demands another extension to the system.

Identify the circuit-board in the toolbox on the workbench

Ben first sits down to figure out what this sentence means. Here is what MM tells Ben: (1) there is a circuit-board in a toolbox (see the relational database above); and (2) that particular toolbox that the circuit board is in is the toolbox on the workbench. So in other words, the sentence says, *Identify the circuit board that is in the toolbox, the toolbox that is on the workbench.*

What would you advise Ben to do now so that the transition tree network will correctly analyze "Identify the circuit-board in the toolbox on the workbench," assuming the meaning given to it? Check just **one** statement below.

☐ You don't have to do anything — the transition tree network, as it emerges from Parts A and B, will handle "Identify the circuit-board in the toolbox on the workbench."

☐ You will have to modify the relational database access routines.

☐ You will have to modify the information provided in the relational database.

☐ You will have to modify the transition tree network, changing `object-description`.

Part D (8 points)

Being an excellent manager, Ben's boss, Alyssa P. Hacker looks at Ben's answer to the problem exposed in Part C and says, "Well, now you can parse a sentence such as, *Identify the circuit board in the toolbox on the workbench*. However, your network will fail in other cases. For example, it will fail to handle sentences such as *Put the circuit board on the workbench* and *Put the circuit-board in the toolbox on the workbench* correctly."

Given the adjustment you recommended, if any, in Part (C), will the network fail as Alyssa says it will?

- Check all the boxes below that explain why the network will or will not fail. Then in the box following the Yes/No questions, explain your choice(s).

☐ Yes, the network will fail, because syntax is never considered by semantic transition trees.

☐ Yes, the network will fail, because semantic transition trees have no facility for backup (called backtracking by some instructors).

☐ Yes, the network will fail, because semantic transition trees allow backup (called backtracking by some instructors).

☐ Yes, the network will fail, because "on the workbench" will be absorbed by the `object-description` network, rather than `main`.

☐ No, the network will not fail because semantic transition trees can figure out what sentences mean without considering syntax.

☐ No, the network will not fail because you can always add transitions that take into account the difference in meaning between two sentences.

☐ No, the network will not fail because semantic transition trees can have arbitrary Scheme procedures attached to them.

☐ No, the network will not fail because the `object-description` network can be defined such that only one place phrase is taken to modify any particular noun.

Your explanation for the box(es) you have checked:

Problem 2 Bayes Nets (31 points)

This problem contains independent parts, so **do not give up on the problem as a whole just because you give up on part of the problem**.

You are given the task of making admissions decisions to MIT. To this end you interview the admissions staff and you conclude that the process can be modeled with four binary-valued random variables:

- Q – indicates whether the applicant is qualified for admission, that is, would inherently be able to do well at MIT.
- G – indicates whether the applicant has a high grade point average in high school.
- R – indicates whether the applicant has excellent recommendations from high school teachers.
- A – indicates whether the applicant is admitted.

You also get the following information from your interviews.

- One third of all applicants are qualified.
- 95% of qualified applicants have a high grade point average. On the other hand, 50% of unqualified applicants also have a high grade point average.
- 90% of qualified applicants have excellent recommendations. On the other hand, 75% of unqualified applicants also have excellent recommendations.
- Applicants that have a high grade point average and excellent recommendations have a 75% chance of being admitted. If an applicant only has a high grade point average or excellent recommendations but not both, then there is a 50% chance of getting admitted. If the applicant has neither a high grade point average nor excellent recommendations, there is no chance they'll get admitted.

Part A (9 points)

We will now construct a Bayes net that captures the data above.

First, in the space below draw *only* the necessary links between the nodes. Do not fill in the probabilities on this picture; we will deal with them later.

Q

G R

A

Now, specify the conditional probability tables for the nodes. In the top row of each table, identify the variables on which the conditional probability depends. Then, use the other rows to show the conditional probability values for each combination of variable values.

We've given you room to write the biggest possible table, you need not use every entry in these tables.

Var Name	Var Name	Var Name	Cond Prob
			P(Q | --,--,--)

Var Name	Var Name	Var Name	Cond Prob
			P(G | --,--,--)

Var Name	Var Name	Var Name	Cond Prob
			P(R | --,--,--)

Var Name	Var Name	Var Name	Cond Prob
			P(A | --,--,--)

Part B (8 points)

Could the conditional probability table for node A (as specified above) have been written, equivalently, using the noisy-or model? If yes, clearly indicate the causal probabilities and leak probability. If no, explain in (at most) two sentences why it cannot be done.

Check YES or NO: ☐ YES ☐ NO

For "yes," give causal probabilities and leak; for "no," give explanation:

Part C (6 points)

The following problems are independent of the ones above.

Given the following Bayes Net:

```
     ---> B
   /
  A
   \
     ---> C ----> D
```

Which of the following is the simplest correct expression for the probability of a fully specified state? Circle your choice.

1. $P(A=a ,B=b,C=c,D=d) = P(D=d|\ A=a,B=b,C=c)\ P(B=b|\ C=c,A=a)\ P(C=c|\ A=a)\ P(A=a)$
2. $P(A=a,B=b,C=c,D=d) = P(D=d|\ A=a,B=b,C=c)\ P(B=b|\ A=a)\ P(C=c|\ A=a)\ P(A=a)$
3. $P(A=a,B=b,C=c,D=d) = P(D=d|\ B=b,C=c)\ P(C=c|\ B=b,A=a)\ P(B=b|\ A=a)\ P(A=a)$
4. $P(A=a,B=b,C=c,D=d) = P(D=d|\ A=a,B=b,C=c)\ P(C=c|\ A=a)\ P(B=b|\ A=a)\ P(A=a)$
5. $P(A=a,B=b,C=c,D=d) = P(D=d|\ B=b,C=c)\ P(B=b|\ C=c,A=a)\ P(C=c|\ A=a)\ P(A=a)$
6. $P(A=a,B=b,C=c,D=d) = P(D=d|\ B=b,C=c)\ P(B=b|\ A=a)\ P(C=c|\ A=a)\ P(A=a)$
7. $P(A=a,B=b,C=c,D=d) = P(D=d|\ C=c)\ P(B=b|\ C=c,A=a)\ P(C=c|\ A=a)\ P(A=a)$
8. $P(A=a,B=b,C=c,D=d) = P(D=d|\ C=c)\ P(B=b|\ A=a)\ P(C=c|\ A=a)\ P(A=a)$
9. $P(A=a,B=b,C=c,D=d) = P(D=d|\ C=c)\ P(B=b)\ P(C=c|\ A=a)\ P(A=a)$
10. $P(A=a,B=b,C=c,D=d) = P(D=d|\ C=c)\ P(B=b)\ P(C=c)\ P(A=a)$
11. $P(A=a,B=b,C=c,D=d) = P(D=d)\ P(B=b)\ P(C=c)\ P(A=a)$

Part D (8 points)

Assume all the variables in the net above are binary valued. Give a symbolic expression for computing P(B=T,D=F | A=T,C=F). Write the expression completely in terms of terms of the form P(A=a,B=b,C=c,D=d), where a,b,c,d are either T or F. **USE THE NUMBER FOR EACH OF THESE TERMS DRAWN FROM THE TABLE BELOW**:

1	P(A=T,B=T,C=T,D=T)	9	P(A=F,B=T,C=T,D=T)
2	P(A=T,B=T,C=T,D=F)	10	P(A=F,B=T,C=T,D=F)
3	P(A=T,B=T,C=F,D=T)	11	P(A=F,B=T,C=F,D=T)
4	P(A=T,B=T,C=F,D=F)	12	P(A=F,B=T,C=F,D=F)
5	P(A=T,B=F,C=T,D=T)	13	P(A=F,B=F,C=T,D=T)
6	P(A=T,B=F,C=T,D=F)	14	P(A=F,B=F,C=T,D=F)
7	P(A=T,B=F,C=F,D=T)	15	P(A=F,B=F,C=F,D=T)
8	P(A=T,B=F,C=F,D=F)	16	P(A=F,B=F,C=F,D=F)

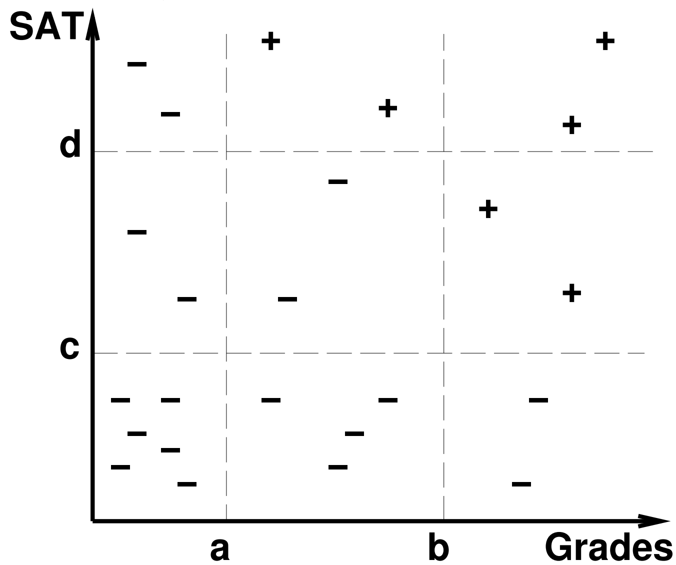
Your answer should involve `ONLY` arithmetic symbols and integers between 1 and 16, as in $\frac{4+12}{7\times 2}$, for example.

Please write neatly!

Problem 3 Identification Trees (20 points)

This problem contains independent parts, so **do not give up on the problem as a whole just because you give up on part of the problem**.

You are given the following admissions data from the MIT admissions office. A "+" indicates someone admitted and a "−" indicates someone not admitted. The x axis is grades (on a 0-100 scale) and the other is average SAT score (on a 0-800 scale):



In building an identification tree for this data, we will consider only the following four tests as candidates for the nodes:

1. **SAT > c**
2. **SAT > d**
3. **Grades > a**
4. **Grades > b**

Part A (4 points)

Compute the value of the average disorder measure for the test **SAT > c**:

Part B (4 points)

Assume that the test **SAT** $>$ **c** is chosen as the test at the top of the identification tree. Consider the remaining tests that could be used at the next level of the tree:

- **SAT** $>$ **d**
- **Grades** $>$ **a**
- **Grades** $>$ **b**

1. Which one of these tests, if any, should be used at the next level of the tree for the branch in which the top-level test (**SAT** $>$ **c**) is *false*. Indicate in one sentence the reason for your answer.

2. Now, for the branch in which the top level test (**SAT** $>$ **c**) evaluates to *true*. Next to each test below, write its rank based on average disorder. That is, write a 1 next to the test with least average disorder, a 2 next to the next best, etc. If there is a tie, use the same rank for the tied tests. **Hint: you need not do detailed calculations to determine the correct answer**.

Test	Rank
SAT $>$ **d**	
Grades $>$ **a**	
Grades $>$ **b**	

Part C (8 points)

Given your success so far, you decide to turn your horizons past college, to the financial future, and use ID trees to work on bankruptcy prediction.

From 10 features (synonym for tests), measured surreptitiously from credit card usage patterns, you propose to a big bank to "guarantee" that you can predict consumer bankruptcies 18 months in advance.

For some reason, the tenth feature, which you thought would be used prominently in the identification tree built by your ID tree program, is not deployed at all for the sample data you provide. Naturally, you suspect a bug, but in fact, when you pay your 6.034 recitation instructor big bucks for consulting, he provides an alternative explanation.

Check all of the following that your recitation instructor might have said, without destroying your faith in his grip on the material. That is, check all of the following that could explain why the tenth feature is not used:

☐ The tenth feature usefully splits the sample data into groups, but it splits the sample data much like the ninth feature, but not quite as well. Therefore, the tenth feature is always dominated by the ninth feature.

☐ The tenth feature is useful, but only in combination with the ninth feature. Because ID tree construction does no look ahead or test combinations, the value of the tenth feature is overlooked.

☐ The values of the tenth feature are all tightly clustered around the mean. The feature can be useful, but you must first divide all values by the standard deviation of the values for the tenth feature in the sample set.

☐ The tenth feature is not useful. However, a new, eleventh feature can be derived from the tenth feature, and that new eleventh feature is useful.

Part D (4 points)

You fail to find much use for this ID program in bankruptcy detection, but you decide to use the same system to do stock selection. Again you start with 20 features, presuming to divide "good" stocks from "bad" stocks.

There are two groups of stocks, Group A and Group B, representing two different company types (such as electronics and health care).

The features are binary. There are about 1000 samples for each of the two groups, Group A and Group B.

For stocks in Group A, you discover that about 10 features are used in just about every path leading from the root node to a terminal node containing "good" stocks or "bad stocks," All 20 features are found in equal proportions in the tree.

For stocks in Group B, you discover that no more than 10 features are used in any path leading from the root node to a terminal node, and in many paths, the number of features is far fewer. The 20 features are found in **un**equal proportions in the tree.

"Hmmm," says your expensive consultant. Reminds me of the overfitting problem that you often see arise in neural networks.

Check the correct answer:

☐ He is more likely to be reacting to the tree built for Group A.

☐ He is more likely to be reacting to the tree built for Group B.

☐ He is equally likely to be reacting either tree.

☐ He is not likely to be reacting either tree.

Problem 4 How they work (23 points)

The purpose of this problem is to test your understanding of various mechanisms at the level required to predict performance in the face of various perturbations.

This problem contains independent parts, so **do not give up on the problem as a whole just because you give up on part of the problem**.

Part A Neural Nets

A.1 (5 points)

You have purchased neural-net training software from MicroMonopoly. Not surprisingly, it is screwed up.

Instead of the *standard* threshold function,

$$l = \frac{1}{1 + e^{-m}}$$

they have provided an *altered* function,

$$l = \frac{1}{1 + e^{+m}}$$

which, when plotted, gives you the mirror image of the *standard* threshold function.

You are told that a network with the *standard* threshold function can be trained to perform a certain recognition task successfully with a given training set.

After a bit of thought, you conclude that there is a set of weights that will enable an identically arranged network to perform the recognition task with the *altered* threshold function.

Next, you learn that the MicroMonopoly training software uses exactly the training formulas given in the book:

$$\begin{aligned} \Delta w_{ij} = &\quad r o_i o_j (1 - o_j)\beta_j, \\ \beta_j = &\quad \textstyle\sum_k w_{jk} o_k (1 - o_k)\beta_k \text{ for nodes in hidden layers,} \\ \beta_z = &\quad d_z - o_z \text{ for nodes in the output layer.} \end{aligned}$$

Check the box next to the best expectation; place an X in the other boxes:

☐ The training software won't work; in principle, networks that use the *altered* threshold function can't be trained using the given training formulas.

☐ The training software may work, but, on the other hand, using initial weights and a rate constant that would be appropriate for the correct threshold function, it may take much longer to train the network, and the training may not converge on a suitable set of weights at all.

☐ Using the initial weights and a rate constant that would be appropriate for the correct threshold function, the software will work just about as it would if there were no bug.

A.2 (5 points)

You call customer "support" and the person who you talk to recommends that you save all files, reformat your disk, and then reload the operating system and all your applications.

Fortunately, before you descend into that hell, "Network 98" is released. Now, the threshold function they provide is:

$$l = \frac{1}{1 + e^{-1{,}000{,}000 \times m}}$$

Check the box next to the best expectation; place an X in the other boxes:

☐ The training software won't work; in principle, networks that use the *altered* threshold function can't be trained using the given training formulas.

☐ The training software may work, but, on the other hand, using initial weights and a rate constant that would be appropriate for the correct threshold function, it may take much longer to train the network, and the training may not converge on a suitable set of weights at all.

☐ Using the initial weights and a rate constant that would be appropriate for the correct threshold function, the software will work just about as it would if there were no bug.

A.3 (5 points)

Finally, you do a work around that enables you to train the network with the following, normal, threshold function:

$$l = \frac{1}{1 + e^{-m}}$$

Your trained network is to be used in a handheld computer game, so you hand the training results over to a circuit designer who burns the network into a VLSI chip.

Unfortunately, the designer works for a MicroMonopoly spinoff. The threshold functions burned into the chip reflect a previous release of MicroMonopoly Network 98:

$$l = \frac{1}{1 + e^{-1000m}}$$

Even more unfortunately, you learn it would cost a fortune to alter most of the characteristics of the chip, but a few things can be done inexpensively.

Check the box next to the right thing to do; place an X in the other boxes:

☐ Multiply all inputs to first-layer nodes by 10^3.

☐ Multiply all weights by 10^3.

☐ Multiply all final-layer outputs by 10^3.

☐ Multiply all inputs to first-layer nodes by 10^{-3}.

☐ Multiply all weights by 10^{-3}.

☐ Multiply all final-layer outputs by 10^{-3}.

☐ None of the above.

Part B Object Recognition (8 points)

Ever more disgusted, you nevertheless buy the MicroMonopoly "Face Recognizer 98" product for your next project, because your manager owns stock in MicroMonopoly.

Face Recognizer 98 does face recognition using Ullman's alignment method described in lecture and reading. Face Recognizer 98 first finds the inside corners of the unknown person's eyes and the top and bottom of the person's nose. Using these feature locations, or a possibly a subset of those features, Face Recognizer 98 predicts where other facial features would be located, given that the unknown is actually the same person as a particular candidate person. Face Recognizer 98 then announces a successful match if those predicted feature locations are observed in the unknown. The software is smart enough to predict no feature locations that would be hidden from view.

The documentation informs you that you need take only one picture for each person you want to recognize. Puzzled, you call MicroMonopoly customer "support" and ask why only one picture is needed.

Check all of the following boxes next to correct responses, in the sense that the conclusion, found after the words "That means," follows from the assumptions, found between the words "We assume" and "That means." Place an X in all the boxes next to incorrect responses:

☐ "**We assume** that humans never tilt their heads; consequently, the problem is a problem of pure rotation of heads around the vertical axis. **That means** we can get by with just one picture."

☐ "**We assume** that our software never confuses one feature with another; hence there is never a feature-correspondence problem. **That means** we can get by with just one picture."

☐ "**We assume** that the picture we have is taken with the person's head in a known, standard orientation (say, the nose points forward and both ears are visible). **That means** we can get by with just one picture."

☐ "**We assume** that human faces are exactly modeled by faces painted on basketballs. Given that constraint, the two-dimensional position of a point, in the usual orthographic projection, tells you the three-dimensional coordinates of that point. From the three-dimensional coordinates of a point, we can determine where the point would be if we rotate the sphere. **That means** we can compute as many pictures as we need for Ullman's recognition scheme from the one picture you supply."

```
;;;; -*- mode: scheme -*- ;;;;

;;;; ATTRIBUTES

(define attribute-list
  ;; Consumes, for example, COLOR SIZE AND WEIGHT
  (define-tree
    '(branch
      ((attribute --> a) (attribute-list --> a-list) rtn (cons a a-list))
      (and (attribute --> a) rtn (list a))
      ((attribute --> a) rtn (list a)))))

(define attribute
  ;; Consumes, for example, COLOR
  (define-tree
    '(branch (color        rtn 'color)
             (colors       rtn 'color)
             (size         rtn 'size)
             (sizes        rtn 'size)
             (weight       rtn 'weight)
             (weights      rtn 'weight))))

;;;; ATTRIBUTE VALUES

(define attribute-value-list
  ;; Consumes, for example, LARGE BLACK AND HEAVY
  (define-tree
    '(branch ((attribute-value --> a-v)
              (attribute-value-list --> a-v-l)
              rtn (append a-v a-v-l))
             (and (attribute-value --> a-v) rtn a-v)
             ((attribute-value --> a-v) rtn a-v))))

(define attribute-value
  ;; Consumes, for example, LARGE
  (define-tree
    '(branch ((size-value --> v) rtn v)
             ((color-value --> v) rtn v)
             ((weight-value --> v) rtn v))))

(define size-value
  ;; Consumes, for example, LARGE
  (define-tree
    '(branch (large        rtn '(size eq? large))
             (medium       rtn '(size eq? medium))
             (small        rtn '(size eq? small))
             (tall         rtn '(size eq? tall))
             (short        rtn '(size eq? short))
             (wide         rtn '(size eq? wide))
             (narrow       rtn '(size eq? narrow))
             (deep         rtn '(size eq? deep))
             (shallow      rtn '(size eq? shallow)))))
```

```
(define color-value
  ;; Consumes, for example, BLACK
  (define-tree
    '(branch (black        rtn '(color eq? black))
             (white        rtn '(color eq? white))
             (blue         rtn '(color eq? blue))
             (red          rtn '(color eq? red))
             (green        rtn '(color eq? green))
             (yellow       rtn '(color eq? yellow)))))

(define weight-value
  ;; Consumes, for example, HEAVY
  (define-tree
    '(branch (heavy        rtn '(weight eq? heavy))
             (light        rtn '(weight eq? light)))))

;;;; OBJECTS

(define object-description
  ;; Consumes, for example, A BIG BLACK HEAVY BLOCK
  (define-tree
    '(branch ((determiner) (object-description --> o) rtn o)
             ((attribute-value-list --> a-v-l)
              (object-description --> o)
              rtn `(db-select ,o '(with ,@a-v-l)))
             ((object --> o) rtn o))))

(define determiner
  ;; Consumes, for example, A
  (define-tree
    '(branch (a) (the))))

(define object
  ;; Consumes, for example, BLOCK
  (define-tree
    '(branch (block     rtn '(db-select blocks '()))
             (blocks    rtn '(db-select blocks '()))
             (box       rtn '(db-select blocks '(with class eq? box)))
             (boxes     rtn '(db-select blocks '(with class eq? box)))
             (ball      rtn '(db-select blocks '(with class eq? ball)))
             (balls     rtn '(db-select blocks '(with class eq? ball)))
             (wedge     rtn '(db-select blocks '(with class eq? wedge)))
             (wedges    rtn '(db-select blocks '(with class eq? wedge)))
             (brick     rtn '(db-select blocks '(with class eq? brick)))
             (bricks    rtn '(db-select blocks '(with class eq? brick))))))

;;;; MAIN TREE

(define main
  (define-tree
    '(branch
      ;; COUNT THE BLOCKS
      (count (object-description --> o)
        if-end-rtn (db-call `(db-report-count (db-count-items ,o))))
```

```
      ;; HOW MANY BLOCKS ARE THERE
      (how many (object-description --> o) are there
        if-end-rtn (db-call `(db-report-count (db-count-items ,o))))
      ;; WHAT COLORS ARE THE BLOCKS
      (what (attribute-list --> a-l) are (object-description --> o)
        if-end-rtn (db-call `(db-show (db-project ,o '(over ,@a-l)))))
      ;; WHAT ARE THE BLOCKS COLORS
      (what are (object-description --> o) (attribute-list --> a-l)
        if-end-rtn (db-call `(db-show (db-project ,o '(over ,@a-l)))))
      ((identify)
         branch
         ;; IDENTIFY THE BRICKS
         ((object-description --> o)
          if-end-rtn (db-call `(db-show ,o)))
         ;; IDENTIFY THE COLORS OF THE BRICKS
         (the (attribute-list --> a-l) of (object-description --> o)
          if-end-rtn (db-call `(db-show (db-project ,o '(over ,@a-l))))))
      ;; RANK THE BRICKS BY COLOR
      ((rank) (object-description --> o) by (attribute-list --> a-l)
       if-end-rtn (db-call `(db-show (db-sort ,o '(by ,@a-l)))))
      (stop if-end-rtn 'quit)
      (rtn (format #t "~%Sorry, I do not understand")
           #f))))

;;;; MISCELLANEOUS

(define (number words)
  ;; Consumes, for example, 3
  (cond ((null? words) #f)
        ((number? (first words)) (list (first words) (cdr words)))
        (else #f)))

(define rank
  ;; Consumes, for example, RANK
  (define-tree
    '(branch (rank)
             (list)
             (sort))))

(define identify
  ;; Consumes, for example, IDENTIFY
  (define-tree
    '(branch (identify)
             (describe)
             (show me all)
             (show me)
             (what is)
             (what are)
             (who is)
             (who are)
             (give)
             (display)
             (print)
             (list)
```

```
              (present)
              (which))))

;;;; ALL CHANGES TO THE ORIGINAL PROBLEM SET 7 GRAMMAR ARE LOCATED BELOW THIS LINE
;;;; "ORIGINAL" MEANS BEFORE THE CHANGES CALLED FOR IN THE PROBLEM SET

(define place
  ;; Consumes, for example, ON THE WORKBENCH
  (define-tree
     `(branch ((place-prep) (object-description --> o) rtn o))))

(define place-prep
  ;; Consumes, for example, ON
  (define-tree
   '(branch (on) (in))))
```