6.034 Quiz 1
October 13, ~~2004~~ 2005

Name	
EMail	

Problem number	Maximum	Score	Grader
1	35		
2	35		
3	30		
Total	100		

Question 1: Rule-based reasoning (35 points)

Mike Carthy decides to use his 6.034 knowledge to take over MIT. To this end, he wants to identify individuals as being communists. He would like to identify less powerful people first, to build a reputation for himself.

He produces the following rule set:

```
(P0
  (IF
   THEN (evaluate-gradstudents)))
(P1
  (IF (evaluate-gradstudents)
       (?person is gradstudent)
       (?person researches ?topic)
       (?topic nationally-valuable)
   THEN (?person communist)))
(P2
  (IF (evaluate-gradstudents)
   DELETE (evaluate-gradstudents)
   ADD (evaluate-professors)))
(P3
  (IF (?x is professor)
       (?y is student of ?x)
       (?y communist)
   THEN (?x communist)))
```

Mike debugs his rule set on the following fabricated data:

```
A1: (Tanis researches finance)
A2: (Jennifer researches ColdFusion)
A3: (Cat researches ethics)
A4: (finance nationally-valuable)
A5: (ColdFusion nationally-valuable)
A6: (Tanis is gradstudent)
A7: (Jennifer is gradstudent)
A8: (Cat is gradstudent)
A9: (Winston is professor)
A10: (Tanis is student of Winston)
A11: (Cat is student of Winston)
```

Part A: Backward chaining (15 points)

Part A.1 (10 points)

You are to simulate backward chaining with the hypothesis (Winston Communist). To do this, you are to fill in the following table with the patterns that the system tries to match against the database, in the order that the system tries to match them against the database. To get started, we have put (Winston Communist) in the table.

~~When writing patterns in the table, write the bindings which are being checked, instead of the variables (i.e. not ?x)~~. You have enough or more than enough room in the table.

Essential Assumptions:

1. Whenever the system's user is asked if an assertion is true, (that is, when the assertion is not found in the assertion database and the assertion does not match any consequent in a rule), the user answers **no.**
2. The backward chainer does not add any assertions to the database, so it can derive the same result multiple times.
3. Conflict resolution is by rule order.
4. If a variable in a rule can be bound to multiple bindings (for example, binding ?topic to finances or ethics), order the bindings according to the order in which the corresponding value appears first in the assertion base.

It may be useful to draw an and-or tree, and we have provided room on a tear-off sheet at the end of the quiz, but we will only grade work shown in the table.

Step	Assertion matched against database	
1	(Winston Communist)	
2	(evaluate - gradStudents)	try P1
3	(Winston is gradStudent)	no. backup
4	(Winston is professor)	try P3
5	(?y is student of Winston)	or node. Try tanis.
6	(tanis communist)	
7	(evaluate - gradStudents)	try P1
8	(Tanis is gradstudent)	
9	(Tanis researches finance)	
10	(finance nationally - valuable)	
11		done. Stop.
12		
13		
14		
15		

Part A.2 (5 points)

Does the backward chainer identify Winston as a communist? Circle your answer:

[✓ Yes] No

Part B: Forward Chaining (10 points)

Mike calls (chain 100) using a forward chainer that uses rule and assertion ordering with the rules and assertions ordered as given above. Who is identified as a communist?

Tanis, Jennifer.

P3 is never executed due to a P0-P2 loop. Hence, Winston is not a Communist. Cat is not researching a "nationally-valuable" topic.

Part C: Genetic Algorithms (10 points)

Mike decides to see if he can improve his rule set using genetic algorithms. He devises an asexual reproduction plan, in which mutations delete rules, and crossovers reorder them.

The fitness of a rule sets is determined as follows:

1. A rule set that doesn't halt in 100 steps of forward chaining get a 0.
2. Otherwise, +1 for every communist identified.

What will Mike need to evolve a rule set which is more fit than the current rule set (gets a higher value from the scoring polynomial, when run on the current assertions)? Circle your answer and explain.

a) mutations

b) crossovers

c) either

d) both

e) neither can make a better rule set on these assertions.

The System loops P0-P2. To stop looping, you must delete ~~[illegible]~~ P2. So, the answer is (a). Crossovers do not help - it will still loop and get a Score of zero.

If you did not See the looping, and indicated this in the text box, the correct answer is e.

Question 2: Search (35 points)

A slider puzzle contains several small, square pieces that fit together into a rectangle with a hole in it. For example:

1	0	0
0	0	■

A move consists of sliding a piece into the empty square. Thus, two moves are possible from the example:

1	0	■
0	0	0

1	0	0
0	■	0

You can solve a slider puzzle by doing a search on it, where every step of the search is sliding one piece to a new place.

If a search uses a heuristic estimate of the distance to the goal, take that heuristic distance metric to be the number of pieces that are in a place that should be blank or occupied by a piece of the other type. For example, if the goal is:

1	■	0
0	0	0

then the heuristic value associated with the following configuration is 2 because a 0 is where a 1 should be and the 1 is where a blank should be.

0	1	■
0	0	0

The actual distance between two states is the number of moves taken to get from the first state to the second.

Break ties according to the position of the blank space when you scan left to right in the top row, then the bottom row as in the following examples:

1	■	0
0	0	0

comes before

1	0	■
0	0	0

which comes before

1	0	0
0	■	0

If two different states remain tied, break that tie using the 1 piece.

Part A: Breadth first search (7 points)

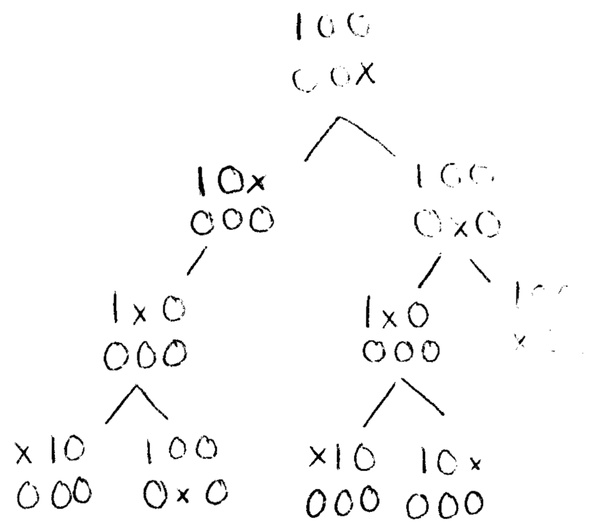
Perform a breadth-first search from

1	0	0
0	0	■

to

1	0	0
■	0	0

, You are to draw the search tree, and each node in the tree is to be identified by 3 x 2 drawing of the puzzle state. Assume that you are using a queue-based algorithm that terminates as soon as a path from the start to the goal reaches the front of the queue. Include in your tree all nodes at the ends of paths that you put on the search queue. Do not use an extended list in your search.

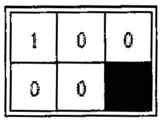


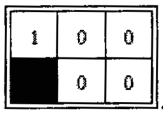
Part B: Depth first search (7 points)

If you were doing a depth-first search, would you need to allow backtracking to ensure that you find a solution for the initial state and goal shown in part A? Circle your answer:

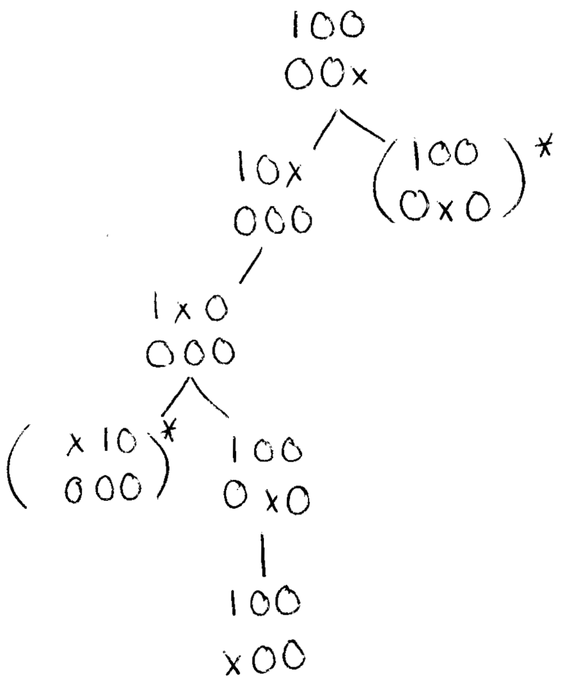
Yes [✓ No]

Part C: Hill climbing (7 points)

Perform a hill-climbing search from 

to 

. As in Part A, draw the tree. Include in your tree all nodes at the ends of paths that you put on the search queue. Do not use an extended list in your search. Do not use backtracking. Use the heuristic distance metric provided in the introduction to this problem.



* Technically, these should not be here, but full credit was given if both were present.

Part D: Admissibility (7 points)

Suppose you define a heuristic to be **admissible** if and only if the heuristic is a lower bound on the number of moves from a given state to the goal state. Given this definition, is the heuristic we have supplied admissible? Circle your answer and explain:

[✓ Yes] No

Because the heuristic is the number of pieces out of place, it must take at least that many moves to get to the goal.

Part E: A* (7 points)

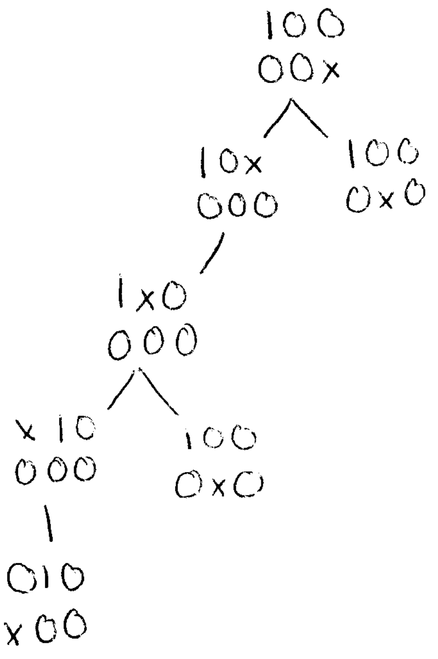
Perform an A* search from

1	0	0
0	0	■

to

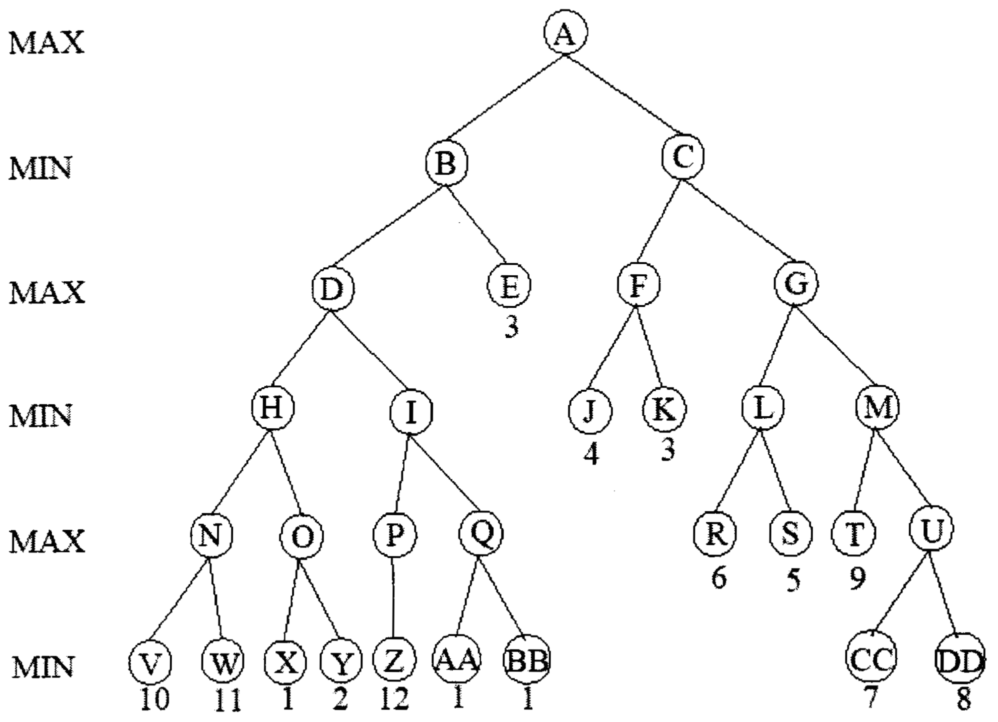
0	1	0
■	0	0

to find the solution to the puzzle that has the minimum number of moves. As in Part A, draw the tree. Note that A* dictates that you must use an extended list. Use the heuristic distance metric provided in the introduction to this problem.



Question 3: Games (30 points)

Consider the following game tree:



The value near a node is the value that would be returned by the static evaluation function if applied to that node.

Part A: Minimax (12 points)

Part A.1 (6 points)

Which is the best choice for the maximizer at node/state A? Circle your answer:

Move to B [✓ Move to C]

Part A.2 (6 points)

What is the minimax value of node A? $\boxed{4}$

+2 , f 2 .

Part B: Alpha beta and progressive deepening (18 points)

Suppose you do not know the best choice of the maximizer at A or its minimax value yet. You are considering using one of the following two methods to compute the best move for the maximizer at A.

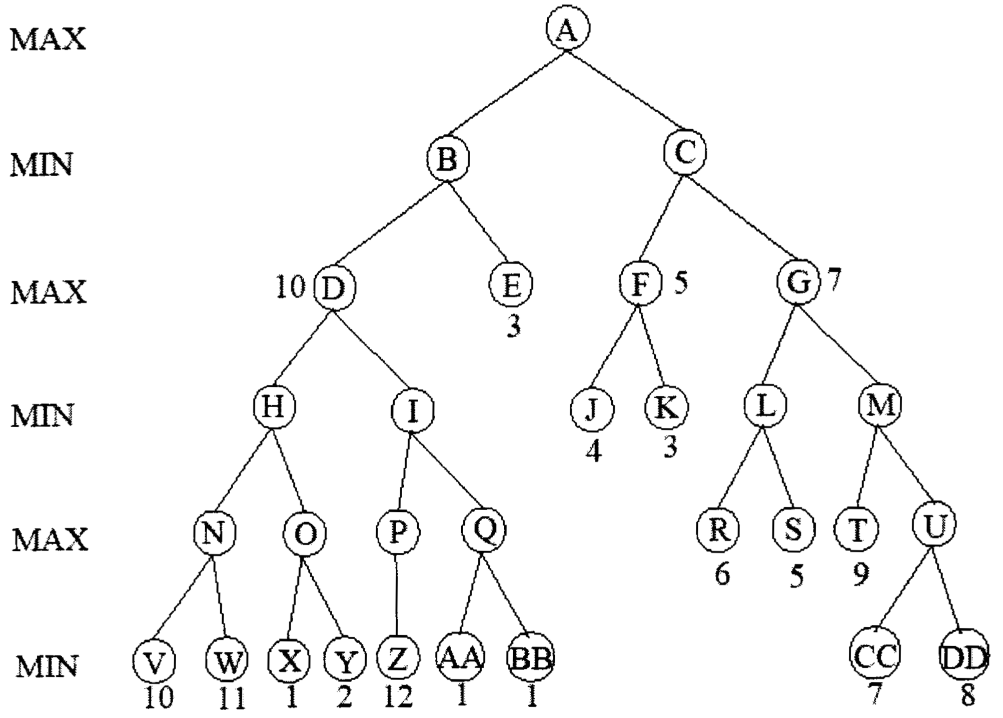
Method A: Alpha-Beta without progressive deeping on the whole tree (i.e., down to depth 5) with a left-to-right order of evaluation for the children of each node.

Method B: A form of progressive deepening with Alpha-Beta. Instead of performing Alpha-Beta down to every depth, you only do it twice: first to depth 2, and then to depth 5.

When running Alpha-Beta down to depth 2, you use a left-to-right order of evaluation for the children of each node and depth 2 static values that have been added to the game tree, as shown on the next page.

Then, when running Alpha-Beta down to depth 5, you **reorder the nodes at depth 1 and depth 2** using the values of the nodes determined while running Alpha-Beta down to depth 2. You reorder so as to try to increase the effectiveness of the full Alpha-Beta search.

Once you reorder the nodes at depth 1 and depth 2, you no longer use the static values at D, F, and G; instead you use only the static values provided at the leaf nodes. For nodes at depth 3 and greater, use the node order as shown in the figure.



Part B1A (1 point)

Show how you would reorder the nodes at level 1 and level two by drawing the rearranged tree down to level 2 (that is, include only nodes A, B, C, D, E, F, and G).

$$\begin{tikzcd}[column sep=small]
 & & & A \arrow[dll, no head] \arrow[drr, no head] & & & \\
 & C \arrow[dl, no head] \arrow[dr, no head] & & & & B \arrow[dl, no head] \arrow[dr, no head] & \\
F & & G & & E & & D
\end{tikzcd}$$

Part B1B (5 points)

Which method requires the smallest total number of static evaluations? To facilitate your work, we have provided extra copies of the game tree at the end of the examination on a tear-off sheet.

Alpha-Beta **without** progressive deepening [✓ Alpha-Beta **with** progressive deepening]

Part B2 (6 points)

Using the method you chose for your previous answer, your method will traverse the tree, extending some nodes and performing static evaluation at others. List the nodes that are extended or statically evaluated in the order that your method does the extension and static evaluation, starting with A.

A B D E C F G A C F J K G L R S B E

A B D H N V W O X Y I P Z Q AA BB E C F J K G L R S

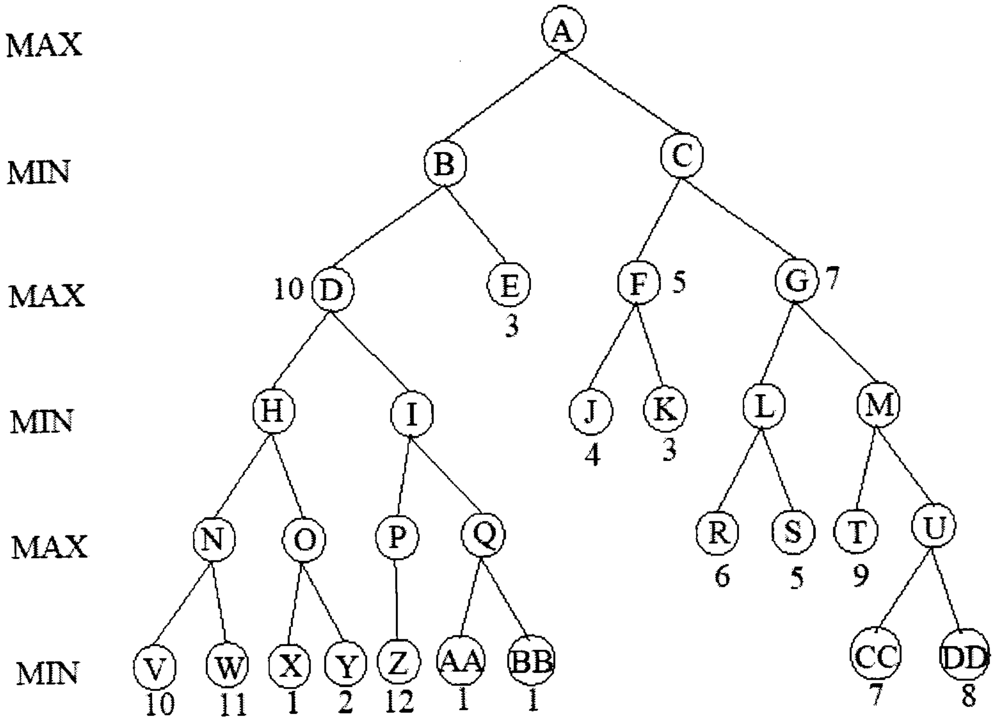
Note: list a node more than once if it is extended more than once while performing the search. For instance, if you chose Progressive *deepening with Alpha-Beta*, node A must appear twice in the list.

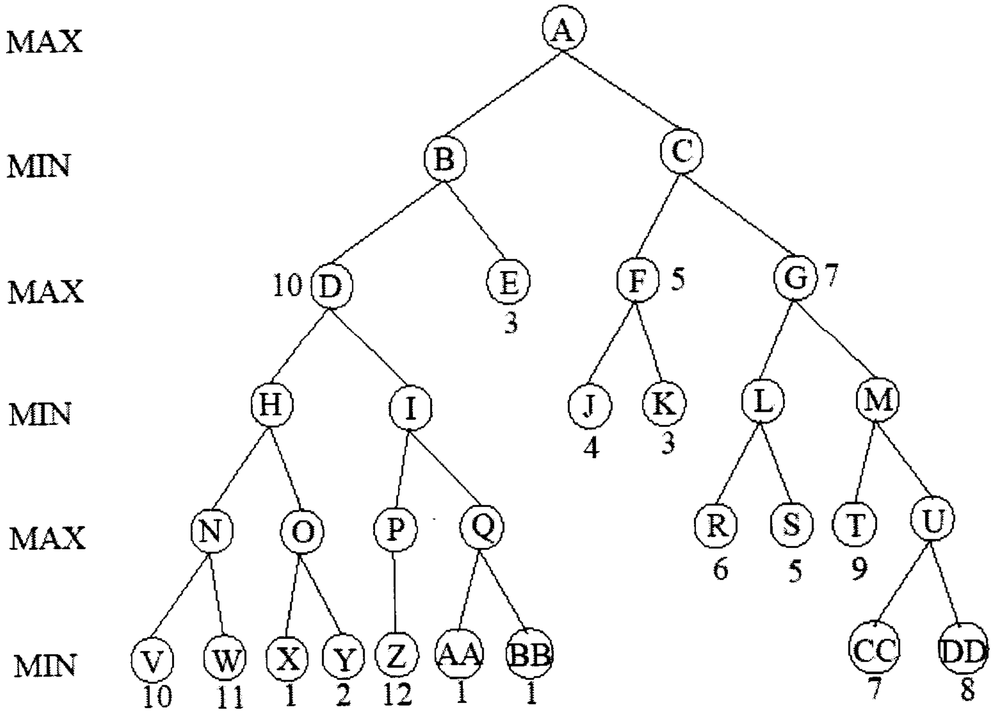
Part B3 (6 points)

How many static evaluations are needed **in total** using your preferred method?

9

12


MAX
MIN
MAX
MIN
MAX
MIN
A
B
C
10 D
E 3
F 5
G 7
H
I
J 4
K 3
L
M
N
O
P
Q
R 6
S 5
T 9
U
V 10
W 11
X 1
Y 2
Z 12
AA 1
BB 1
CC 7
DD 8


MAX
MIN
MAX
MIN
MAX
MIN
A
B
C
10 D
E 3
F 5
G 7
H
I
J 4
K 3
L
M
N
O
P
Q
R 6
S 5
T 9
U
V 10
W 11
X 1
Y 2
Z 12
AA 1
BB 1
CC 7
DD 8

Scratch paper for drawing trees, etc.