Instructor: Russ Greiner

Cmput 366

Final Examination

GIVEN: Wednesday, 13 December 2000

COVERING: Entire course:

This is a closed-book exam; you are allowed only a 1-page "cheat sheet". Calculators are not needed. Please write your answers on these pages.

This test is worth 36% of the course grade.
Time: 180 minutes

You should have 17 pages (including this one).

Name (underline surname):

Student number:

E

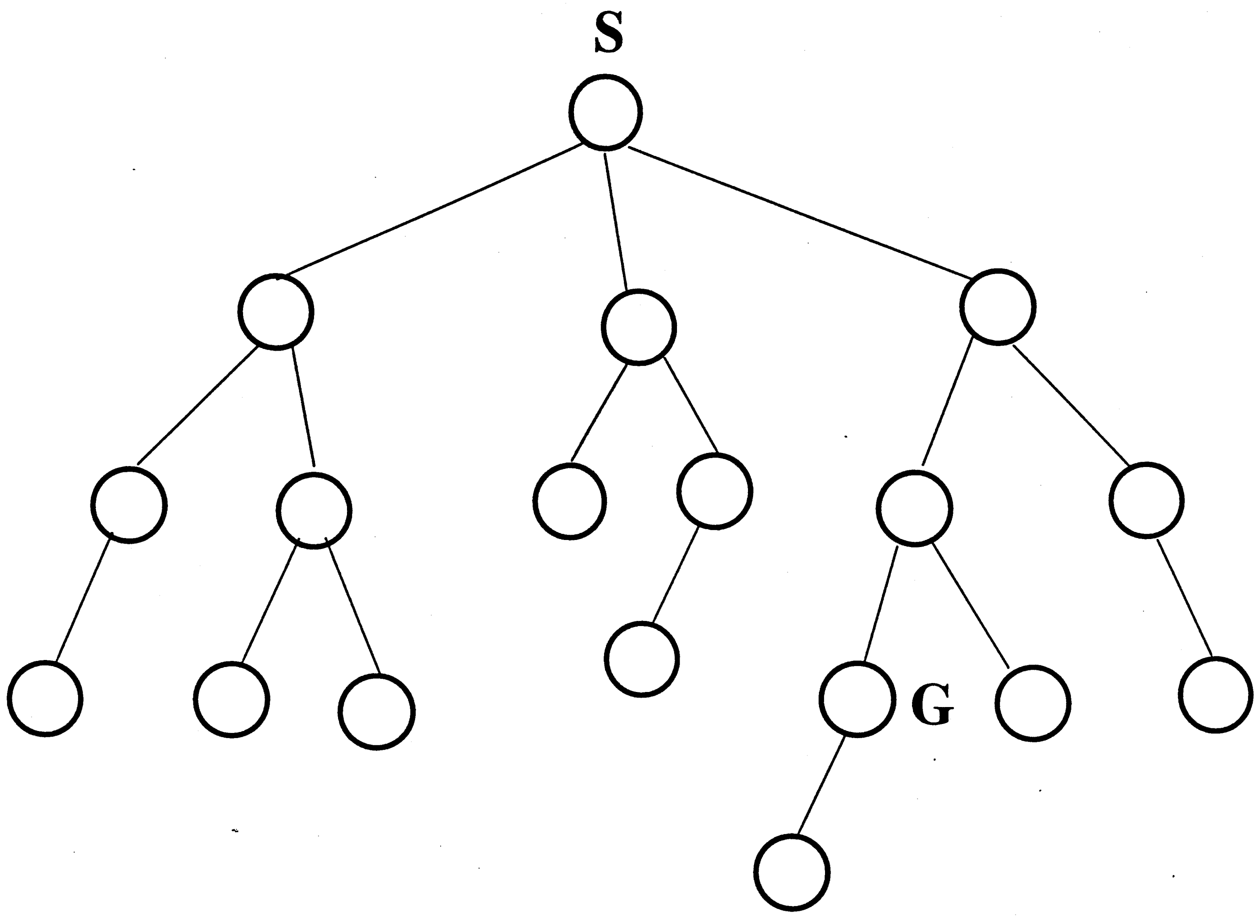
05184
CMPUT 366
GREINER, R.
DEC 00 FINAL
PAGES: 17

1	/ 14
2	/ 10
3	/ 7
4	/ 12
5	/ 8
6	/ 17
7	/ 15
8	/ 5
9	/ 10
10	/ 10
11	/ 12
Total:	/ 120

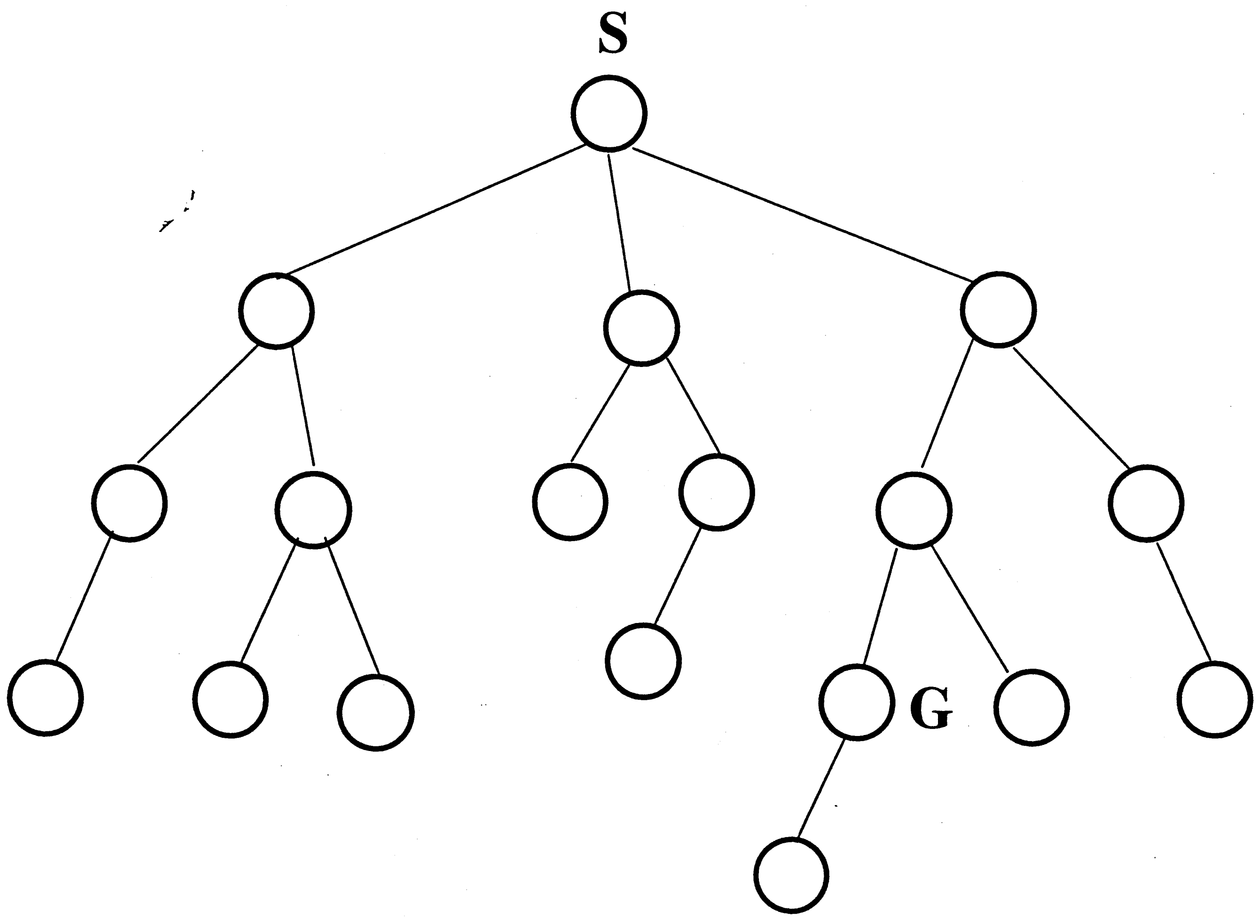
Good luck!

Problem 1 *[14 points]* **Search**
In this problem, you will implement different search strategies on a given search tree. The start state is denoted by S and the goal state by G. Number the nodes in the tree according to the order in which they will be expanded. (Recall that a node is expanded when it is removed from the list of nodes, checked for "goalness", and its children are inserted into the list.) Do not number a node if it is not expanded in the search. Assume that the children of a node are inserted into the list in left to right order, and that nodes of equal priority are extracted from the list in FIFO order. Write your answers directly on the trees provided below.

a [1]: *Breadth First Search*

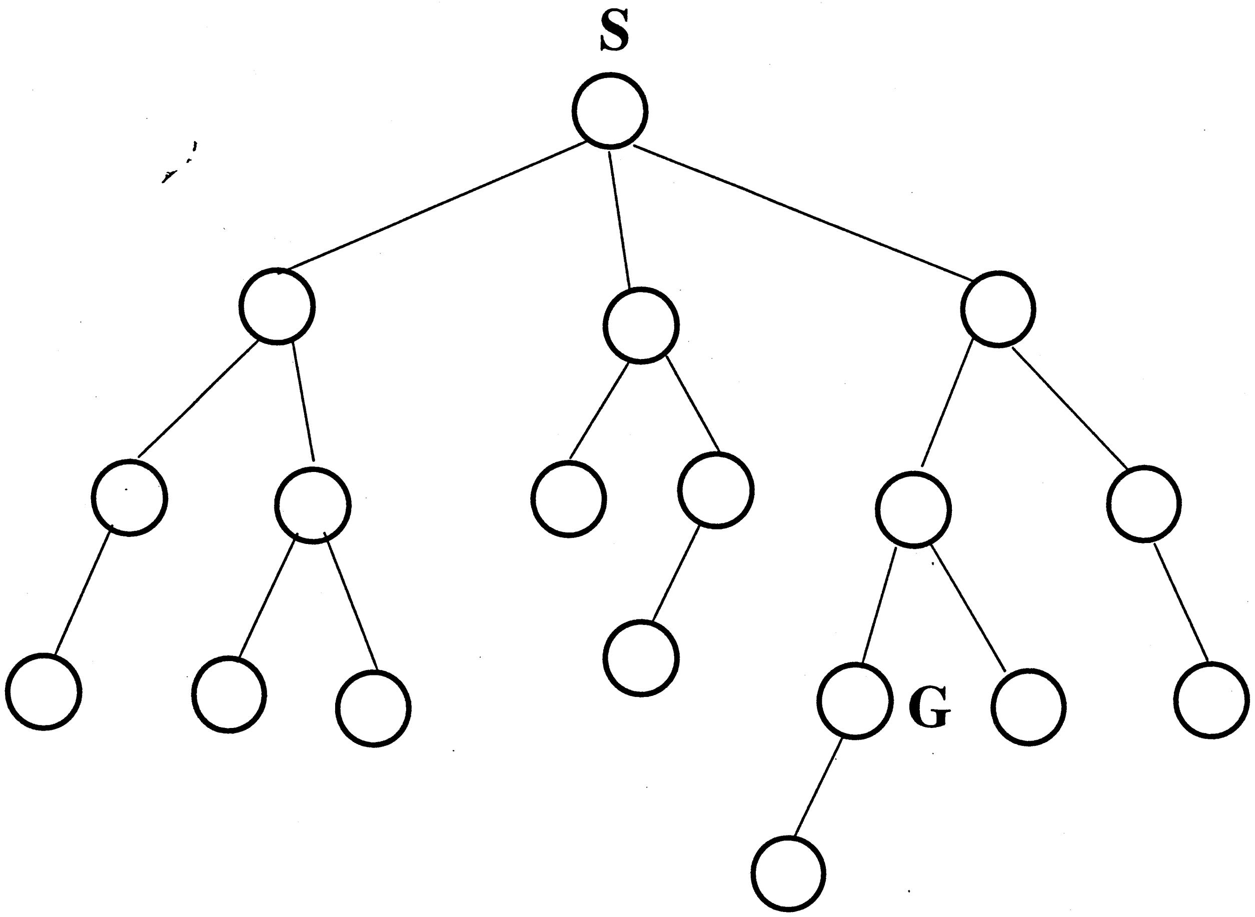


b [1]: *Depth First Search*

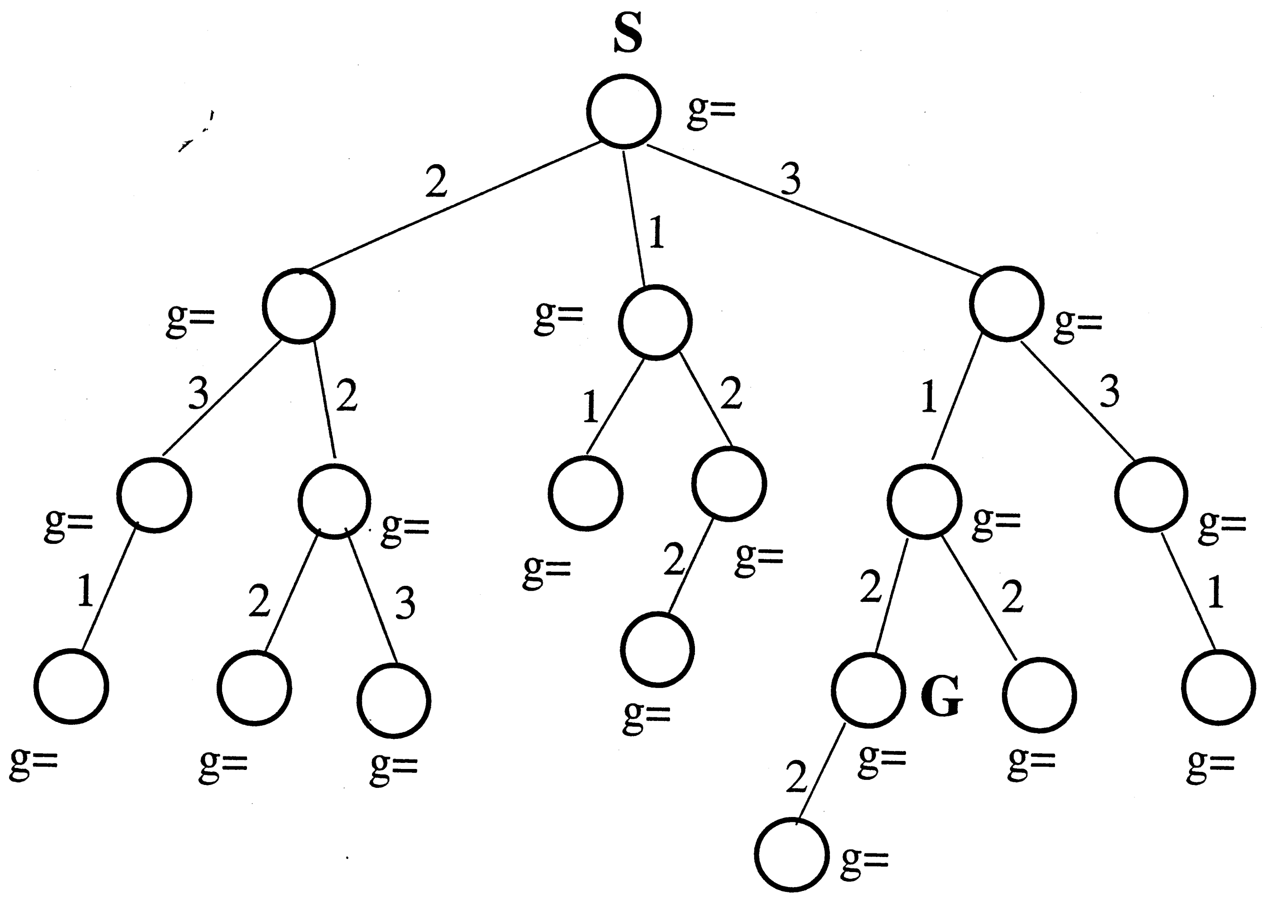


c [3]: *Iterative Deepening*

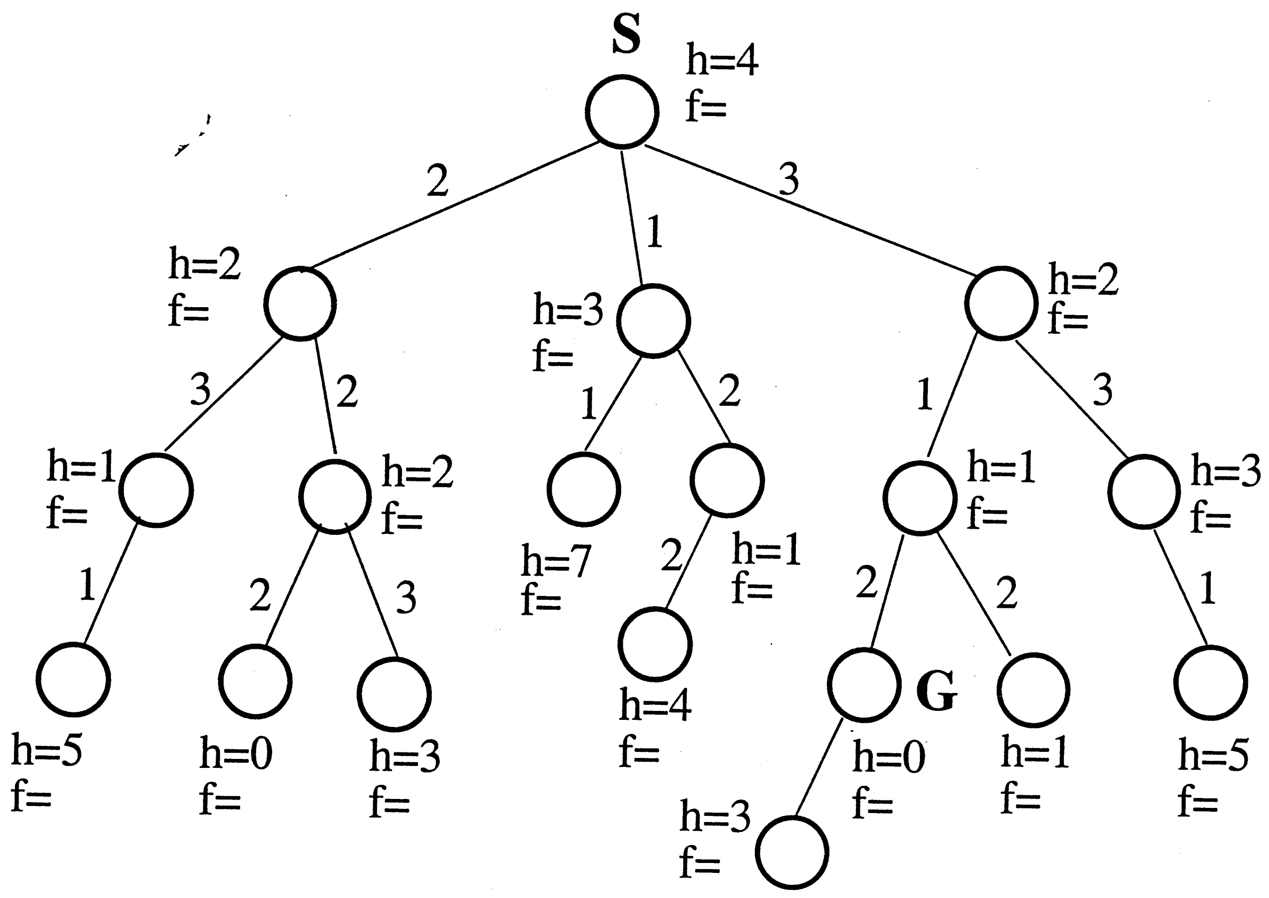
[Hint: Here, a node may have multiple labels.]



d [4]: *Uniform Cost Search*, where the costs of the edges are as specified on the tree. Here, also write down the g-value of the different nodes in the tree. You only need to write down g-values for nodes that are inserted into the list.



e [5]: A^*, where the h-value of each node and the costs of the edges are as specified on the tree. Here, also write down the f-value of the different nodes in the tree. You only need to write down f-values for nodes that are inserted into the list.



Problem 2 *[10 points]* **Constraint Satisfaction:**
A Latin Square is an $N \times N$ array filled with colors, in which no color appears more than once in any row or column. Finding a solution to the following 4×4 Latin Square has been formulated as a CSP, with a variable for each cell in the array, each having a domain of $\{r, g, b, y\}$, and a set of constraints asserting that

⋆ any pair of cells appearing in the same row must have different colors, and
⋆ any pair of cells appearing in the same column must have different colors.

(Note diagonals *can* have repeats.)

At this point in the search, seven of the cells have been instantiated (displayed in boldface), and the initial domains of the remaining cells are shown. As part of the process of instantiating this next cell, indicate which domain values in which squares can be eliminated using our various CSP inference techniques. Indicate eliminated values by crossing them out.

a [3]: *Consistency-checking.* For each of the 9 currently uninstantiated cells, indicate which values would be eliminated, assuming it was the next cell to be processed.

r	**b**	**g**	**y**
g	**r**	r g b y	r g b y
b	r g b y	r g b y	r g b y
r g b y	r g b y	r g b y	r g b y

b [7]: *Arc-Consistency (+ ForwardChecking, Consistency-Checking).* Show the effects of applying arc-consistency throughout *all* of the remaining uninstantiated cells, based on all the currently instantiated cells. Don't forget to eliminate a value in one cell, if that value implies some other variable will have no remaining legal values.

r	**b**	**g**	**y**
g	**r**	r g b y	r g b y
b	r g b y	r g b y	r g b y
r g b y	r g b y	r g b y	r g b y

Problem 3 *[7 points]* **GSAT and Stochatic Search**
Here, we analyze the GSAT algorithms for (attempting to) solve SAT problems, using a greedy hill-climbing algorithm. Each state corresponds to a complete assignment. The successor operator *Succ(s)* generates all neighboring states of s, which we will define as all total assignments which differ by one. So for example given the state with total assignment $\{A \leftarrow \text{True};\ B \leftarrow \text{False}\}$, the neighboring states would be $\{A \leftarrow \text{False};\ B \leftarrow \text{False}\}$ and $\{A \leftarrow \text{True};\ B \leftarrow \text{True}\}$. The value of any state is the number of clauses that are satisfied, given the assignment of the state.

a [1]: If you have n variables, how many neighboring states does the *Succ(s)* function produce?

b [1]: What is the total size of the search space over n variables — *i.e.*, how many possible states are there total?

c [3]: Consider the following set of clauses:

$$\{\neg A \vee B \vee C\},\quad \{A \vee \neg B \vee C\},\quad \{A \vee B \vee \neg C\},\quad \{A \vee B \vee C\}$$

Below show a non-goal state (*i.e.*, a non-satisfying assignment) that is on a plateau in our hill-climbing space; also briefly explain why this is a plateau.

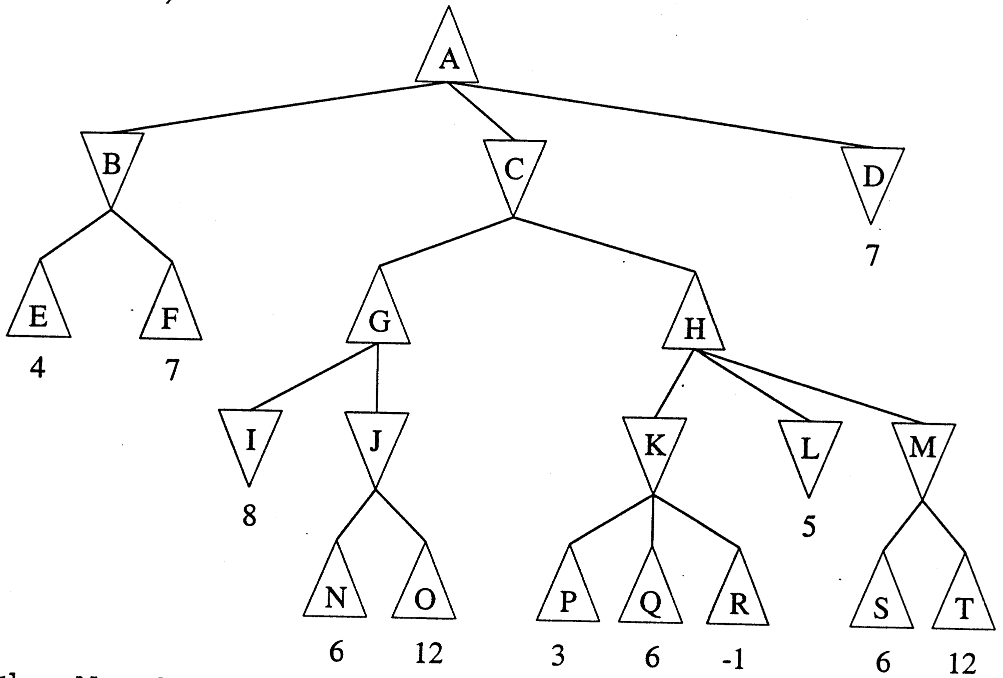
A =
B =
C =

d [2]: Describe a general class of situations where the GSAT stochastic search process is *not* guaranteed to terminate and produce the appropriate answer.

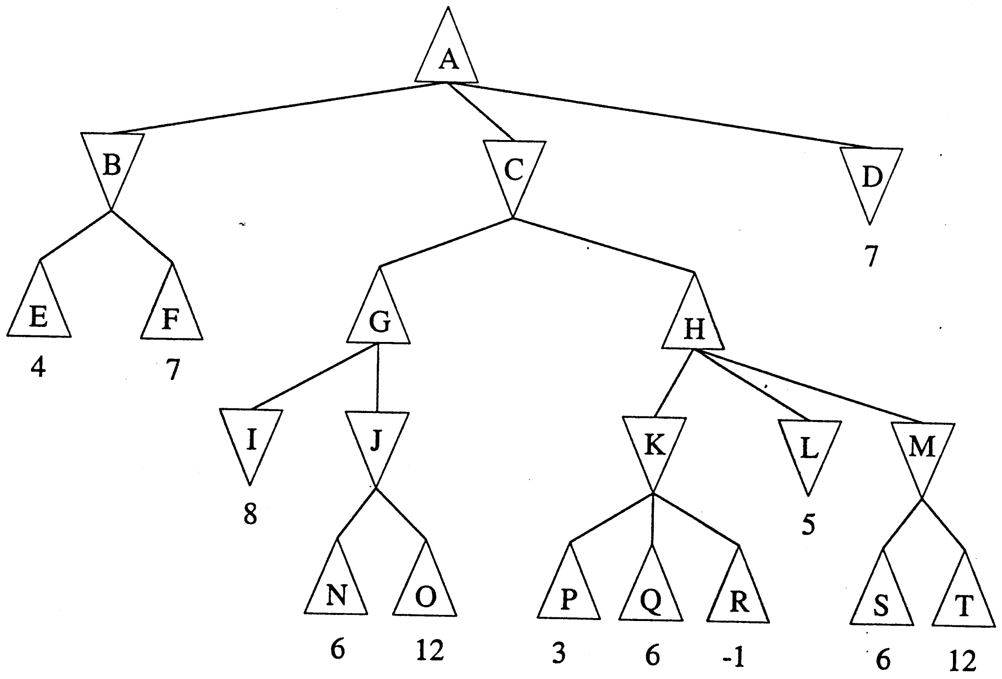
Problem 4 *[12 points]* **Game Playing Search**

a [2]: In the tree below, fill in the values for the internal nodes using the given values at the leaves and the Minimax procedure. (Write the values on the first tree below.)

b [5]: Then perform AlphaBeta pruning on this tree, assuming lefttoright node expansion. Indicate which nodes would be pruned by AlphaBeta pruning by crossing-out the nodes which do not need to be examined. The root node is the *maximizing* player.



c [5]: Now show the pruning, assuming a *right-to-left* order of node expansion.



Problem 5 *[8 points]* **Inference Process**

Consider an inference process $\vdash_{Try100}$ that, given a knowledge base Σ and a query τ, performs only (at most) 100 derivations on Σ (and on conclusions reached so far) before terminating. If τ appears in the original list of Σ's sentences, or in the list of sentences computed, then $\vdash_{Try100}$ returns `Yes`. If not, $\vdash_{Try100}$ returns `No`.

(In answering each of the following questions, be sure to give a justification — which probably means indicating what the term (*e.g.*, "sound") means, and if appropriate, providing a counter-example.)

a [2]: Is this $\vdash_{Try100}$ process "sound"?

b [2]: Is this $\vdash_{Try100}$ process "complete"?

c [2]: What does the response "`No`" mean here? In particular, should you believe $\Sigma \models \neg\tau$?

d [2]: Can we view this reasoning system as making the "Closed world Assumption"? Can we view this reasoning system as making the "Unique Names Assumption"? Explain.

Problem 6 *[17 points]* **Resolution Proof + FirstOrderLogic**
Consider the following facts:

- In every program there is a bug.
- A program that contains a bug does not work.
- Q is a program.

a [3]: Translate these sentences into formulas in first order logic, using the predicates:

`Program(p)`	true if p is a program
`HasBug(p)`	true if the program p has a bug
`Works(p)`	true if the program p works.

b [2]: Convert the formulas into disjunctive normal form.

c [5]: Prove that `Q` does not work, using *refutation resolution.*

d [5]: Here, you are required to rewrite the same information (from part (a)), but using a different vocabulary. In particular, you should NOT use the `HasBug` predicate but instead, you should use

`Bug(x)`	true if the body of code `x` is buggy
`Contains(p, x)`	true if the program `p` contains the code `x`

as well as `Program()` and `Works()`. Now rewrite the claim

A program that contains a bug does not work.

in this vocabulary, expressed in CNF.

e [2]: Circle all that are correct, and *briefly* explain your answer.
A set of formulae $\{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ is not consistent, if and only if:

1. the sentence $\neg\sigma_1 \wedge \neg\sigma_2 \wedge \cdots \wedge \neg\sigma_n$ is satisfiable.
2. the sentence $\neg\sigma_1 \wedge \neg\sigma_2 \wedge \cdots \wedge \neg\sigma_n$ is valid.
3. the sentence $\neg\sigma_1 \vee \neg\sigma_2 \vee \cdots \vee \neg\sigma_n$ is satisfiable.
4. the sentence $\neg\sigma_1 \vee \neg\sigma_2 \vee \cdots \vee \neg\sigma_n$ is valid.

Problem 7 *[15 points]* **Planning, Situation Calculus**

In these days of broken feet and other maladies...

Below we describe some actions which lead to healthy and unhealthy situations. You should restate them more formally — first using *STRIPS*-style actions, and then situation calculus.

> You become *unhealthy* (at time $T+1$) if, at time T,
> your visit a person who is currently unhealthy or
> you are clumsy while at the basketball court
> (as this means you trip while playing basketball)
>
> You become *healthy* (at time $T+1$) if, at time T,
> you have "magic medicine" (which you eat)
> (Note this magic medicine cures any and all unhealthinesses, and works even if you start healthy.)

And, if you perform any other action — like teach a class, order a pizza, tripping while NOT playing basketball, etc. — your state of (un)health is unchanged.

a [10]: Write down the *STRIPS*-ish encoding of the three actions:
`VisitSickie(x)`, `BBTrip(x)`, `TakeMMedicine(x)`
by specifying the preconditions and the effects. You should use the predicates

- `HState(x, h)` means person `x` is in health-state `h`, where `h` $\in$ {`Healthy`, `UnHealthy`}
- `Visit(x, y)` means person `x` visits person `y`
- `AtBBCourt(x)` means person `x` is at the basketball court
- `Clumsy(x)` means person `x` is clumsy
- `HasMM(x)` means person `x` has magic medicine

(Remember to indicate when the effects of an action cause a predicate to NOT hold, by using ¬predicate. Note you cannot be both healthy and unhealthy.)

b [5]: Now write one successor-state axiom associated with `HState(x, Health, do(a, s))`. (Note you do NOT have to write one for `HState(x, UnHealth, do(a, s))`.) You may assume that the actions listed above are the only ones that can affect one's health, but you may NOT assume that these are the only possible actions. You should use the notation `do(a, s)` to denote the situation produced by performing action a in situation `s`; and use predicates of the form `HState(x, Health, s)`, to indicate that the `HState` of `x` is `Health` in situation `s`, etc.

Hint: Recall the performing an action when the preconditions are not met, does not lead to its effect.

Problem 8 *[5 points]* **Learning**

You have two learning algorithms L_1 and L_2 for learning classifiers (think "decision trees"). Given a sample S of 1000 "labeled instances", of the form

$$S = \left\{ \begin{array}{cccc|c} x_1 & x_2 & \cdots & x_n & y \\ \hline 1 & 1 & \ldots & 0 & 1 \\ 1 & 0 & \ldots & 0 & 1 \\ 0 & 0 & \ldots & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & \ldots & 0 & 1 \end{array} \right\}$$

L_1 produces the classifier D_1, and L_2 produces the classifier D_2, each of which maps $\{1,0\}^n \mapsto \{1,0\}$. You observe that D_1 gave the correct answer to 950 of the 1000 instances in S, while D_2 was correct 935 times.

You then get an additional set of 100 new labeled instances, S'; and here observe that D_1 was correct 91 times and D_2 was correct 96 times.

You now have to commit to either D_1 or D_2. Which would you prefer — that is, which do you think will have better accuracy in the future? Explain.

Problem 9 *[10 points]* **Probability**
As a contestant on a game show, you are given the choice of four closed boxes. Inside one is a diamond (yeah!), inside the other three, lima beans (yuck!); unfortunately you don't know which is which. You randomly select a box, say number 1. The host, who knows what's inside the boxes, opens another box, say number 4, which has lima beans. He then asks, "Do you want to abandon box 1, or instead, select another box? If so, you may decide either 2 or 3."

a [7]: Should you accept this offer, and receive the contents of (say) box 2; or keep your original selection of box 1. Explain your answer carefully.

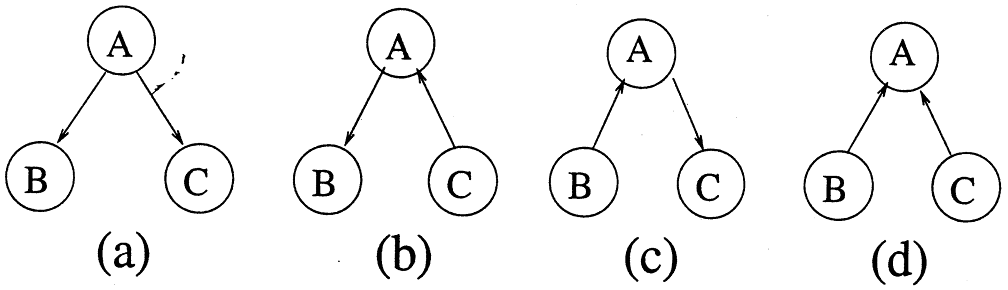
Here you may assume the host *always* offers you the chance to switch boxes.

b [3]: Now suppose the host gets to decide whether to offer you this choice. Does this affect your answer?

Problem 10 *[10 points]* **Belief Nets; I-Equivalence**

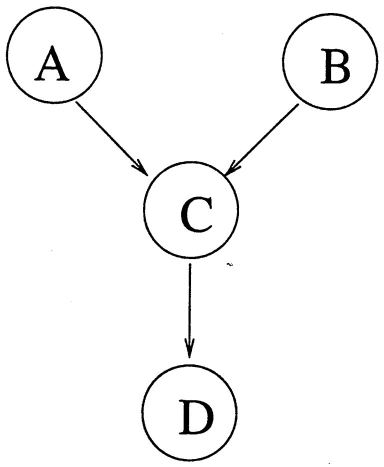
Recall that two belief networks over the same variables, bn_1 and bn_2, are said to be I-*equivalent* if all d-separation properties of bn_1 also hold for bn_2 and vice versa.

a [3]: There are four networks shown below. Which ones are I-equivalent to one another?

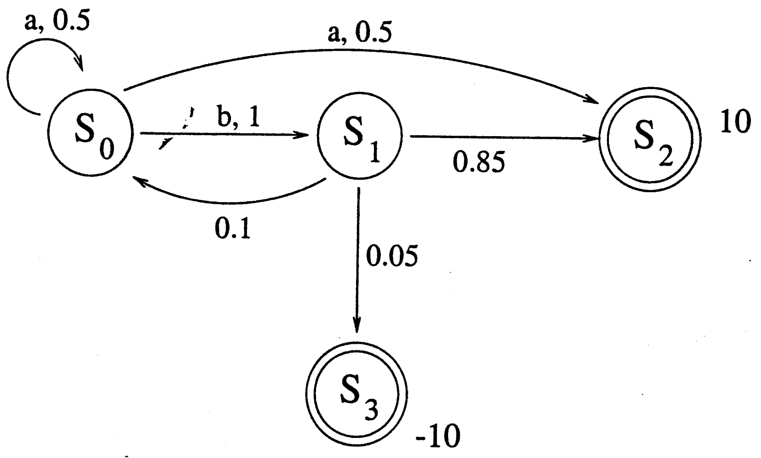


b [7]: In the network pictured below, enumerate all of the conditional independencies represented by the network. Each independence should be of the form $I(X, Y|Z)$ — read "X is independent of Y, given Z — where X, Y and Z are each sets of variables, where Z might be empty.

If your list contains $I(X, Y \mid Z)$ then it need not contain $I(X', Y \mid Z)$ where $X' \subset X$ and $X' \neq X$, nor need it not contain $I(Y, X \mid Z)$.



Problem 11 *[12 points]* **Markov Decision Processes**
Consider the following MDP, where S_2 and S_3 are each terminal states, and there is only one action for S_1 (which happens to be stochastic). The rewards are $R(S_0) = R(S_1) = 0$, $R(S_2) = 10$, $R(S_3) = -10$. You can ignore the discount factor — *i.e.*, $\gamma = 1$.



a [6]: The "π_b" policy takes action b in state S_0. Calculate the associated value function $U_0(S_i) = U^{\pi_b}(S_i)$ for all states S_i.

b [6]: Determine the policy π_0 that is best for these $U_0(S_i)$ values.