Name:_____

0	1	2	3	4	5	6	7	8	total
0	*20*	*20*	*15*	*15*	*25*	*25*	*30*	*30*	*180*

UMBC CMSC471/671 Midterm Exam
October 16, 2006

Please write all of your answers on this exam. The exam is closed book and consists of 8 problems which add up to 180 points. You have the entire class period, seventy five minutes, to work on this exam. Good luck.

0. Warm Up Exercise (0)

In a rectangular array of robots, which will be smartest: (a) the smartest of the dumbest robot in each column, or (b) the dumbest of the smartest robot in each row? (Hint: Save this one for last)

The dumbest of the smartest robot in each row will be smarter than, or equally intelligent as, the smartest of the dumbest robot in each column. There are four cases.

- **The first is that the dumbest of the smartest and the smartest of the dumbest are the same robot, so obviously in this case the dumbest of the smartest and the smartest of the dumbest would have the same IQ.**
- **The second case is that the dumbest of the smartest and the smartest of the dumbest are in the same row. The dumbest of the smartest robot in each row is obviously the smartest robot in his row, so he's smarter than the smartest of the dumbest, who is also in his row.**
- **The third case is that the dumbest of the smartest and the smartest of the dumbest are in the same column. The smartest of the dumbest robot in each column is obviously the dumbest robot in his row, so he's dumber than the dumbest of the smartest, who is also in his column.**
- **The fourth case is that the dumbest of the smartest is neither in the same column nor the same row as the smartest of the dumbest. For this case, consider the robot X who is standing in the intersection of the row containing the dumbest of the smartest and the column containing the smartest of the dumbest. X must be smarter than the smartest of the dumbest, since the smartest of the dumbest is the dumbest in his column, and X must also be dumber than the dumbest of the smartest, since the dumbest of the smartest is the smartest in his row. So smartestOfDumest < X < sumbestOfSmartest.**

So the dumbest of the smartest in each row is always smarter than, or equally smart as, the smartest of the dumbest in each column.

1. Unification (20)

Each of the following Prolog expressions invokes unification. For each, say whether or not the unification would succeed and, if it does, what values the variables in the terms would take on. Assume that the same variable name in two different terms represents the same variable.

a) `ancestor(X,Y,[]) = ancestor(john,mary)` **FAILS: different arity**

b) `Rel(X,mary) = parent(john,Y)` **FAILS: predicate can't be a variable**
c) `loves(john, mary) = loves(Mary, john)` **FAILS: john ≠ mary**
d) `husband(John, Mary) = wife(Mary, John)` **FAILS: predicates don't match**
e) `spouse(John, Mary) = spouse(father(Bill), mother(Bill))` **succeeds with John=father(Bill), Mary=mother(Bill)**
f) `father(mother(bill)) = father(Jane)` **succeeds with Jane=mother(Bill)**
g) `child(Mary, mother(Bill)) = child(child(Bill), Mary)` **FAILS: mother(Bill) ≠ child(Bill)**

h) `child(child(Sally), mother(Bill)) = child(child(John), Sally)` **succeeds with Sally=John=mother(Bill)**
i) `[ [] | [] ] = [ X ]` **succeeds with X=[]**
j) `[X, Y] = [john, mary | Z]` **succeeds with X=john, Y=mary, Z=[]**

2. True/False (20 points)

Circle either T or an F in the space before each statement to indicate whether the statement is true or false. If you think the answer is simultaneously true and false, quit while you are ahead. There is no penalty for incorrect answers (but then, there are no points for incorrect answers either).

T F The Turing test evaluates a computer system's ability to act rationally. **FALSE**
T F The ultimate goal of AI is to design systems that can think and act like humans. **FALSE**
T F A stochastic environment is one in which the next state is completely determined by the agent's action. **FALSE**
T F Prolog variables do not have a declared type. **TRUE**
T F Prolog lists are built using terms of arity two and a principle functor (i.e., predicate name) '.'. **TRUE**
T F The only reason that Prolog is not considered a "pure logic programming language" is how it does numerical computations. **FALSE**
T F A simple breadth-first search will always find the shortest solution if one exists and it is of finite length. **TRUE**
T F A danger of depth-first search is that it may not terminate if the search space is infinite, even if a finite solution exists. **TRUE**
T F Algorithm A* will always find the shortest solution if one exists and it is of finite length. **TRUE**
T F Iterative Deepening search is a practical way to add heuristics to algorithm A. FALSE
T F Algorithm A will perform a depth-first search if f(n) = - g(n). **TRUE**
T F An advantage of Hill Climbing search is that it requires minimal memory. **TRUE**
T F Simulated annealing search can avoid being stuck on local maxima that are not true solutions. **TRUE**
T F Bi-directional search is always more efficient than uni-directional search. **FALSE**
T F The prisoner's dilemma shows that game theory is not sound. **FALSE**
T F Game theory applies only to zero-sum games. **FALSE**
T F Forward checking is a more powerful constraint propagation algorithm than arc consistency in that it can converge to a solution more quickly. **FALSE**
T F The minimax algorithm has been adapted to games that include a chance element, such as backgammon. **TRUE**

T F The alpha-beta algorithm is preferred to minimax because it provides a better estimation of which move is best for a given lookahead distance. **FALSE**

T F Deciding if a CSP is consistent is, in general, NP-hard. **TRUE**

3. Mystery predicate (15)

(a) Describe in a few sentences what the *foo/2* predicate does. (b) Give several examples to illustrate its intended use. (c) Describe any requirements for its arguments, e.g., must any be instantiated? Must they be of any type?

```
foo([],1).
foo([H|T],N) :- foo(T,M), N is M*H.
```

(a) foo(+List,-N) is true of List is a list of numbers and N is their product.
(b) foo([1,2,3,4],24) succeeds, foo([7,3,4], X) succeeds and unifies X with 84, foo([8],8) succeeds.
(c) The first argument must be instantiated to a list of numbers and the second can be a variable or a number.

4. Prolog between/3 (15 points)

Write the prolog predicate *between(+From,+To,?X)* which is true if *X* is an integer between the integers *From* and *To*. Assume the first to arguments are valid integers, *From* ≤ *To*, and the third is either an integer or a variable. For example:

```
?- between(1,10,3).          ?- between(1,10,11).
yes                          no
?- between(1,3,X).           ?- between(2,1,X).
X=1;                         no
X=2;                         ?- between(3,3,X).
X=3;                         X=3;
no                           no
```

.

between(N,M,N) :- N=<M.
between(N,M,X) :- N=<M, N2 is N+1, between(N2,M,X).

5. Nim2x2 (25)

Nim2x2 is a game for two players, whom we will call A and B. Initially, there are two heaps, each containing two stones. The players alternate turns, and at each the current player removes a non-zero number of stones from either heap. At least one stone must be removed during a turn. The last player to remove stones loses. Player A makes the first move. A simple representation of the board might be a tuple (N,M,P) where N and M are the number of stones in the first and second heaps, respectively, and P designates the player: A or B. So the initial state would be (2,2,A).

(a) [2] Show the board representation(s) for a win for player A. **(0,0,A)**

(b) [2] Show the board representation(s) for a win for player B. . **(0,0,B)**

(c) [12] Draw the complete game tree for Nim2x2. Show this as a tree, so states may be repeated. **See attached**

(d) [12] Use a static evaluation function that assigns a win for player A the value 1 and a win for player B the value -1 and all other boards the value 0. Compute the minimax values for each of the states in the tree shown in your game tree, writing the values next to the nodes. **See attached**

(e) [2] If both players play perfectly, what will be the outcome; will it always end in a draw, a win for A, a win for B, or is it impossible to tell? **It is always a win for B.**

6. Algorithm A (25)

Consider a search space defined by the following table, which gives the cost of arcs between pairs of nodes. Node A is the start state and node Z the goal.

from	A	A	A	B	C	D	E	F	G
to	B	C	D	E	E	F	G	G	Z
cost	3	2	4	3	5	1	2	4	3

(a) [5] Draw the complete graph of the state space. **See attached**

(b) [5] What path is the cheapest solution and what is its cost?

The shortest path is A, B, E, G, Z with cost 11.

(c) [5] Using graph search with a selection function of **f(n) = g(n)**, in what order would the nodes in the graph be expanded. Recall that expanding a node means computing its successors and adding them to the graph or adding links to them if they are already present in the graph. Recall that g(n) is the cost of the cheapest known path from the start node to n.

node	A	B	C	D	E	F	G	Z
order	**1**	**3**	**2**	**4**	**6**	**5**	**7**	**8**

(d) [5] Repeat this exercise using the selection function **f(n) = -g(n)**.

node	A	B	C	D	E	F	G	Z
order	**1**			**2**		**3**	**4**	**5**

(e) [5] Assuming the following values for the heuristic function h(n), show the order that the nodes would be expanded if the selection function is **f(n) = g(n) + h(n)**.

node	A	B	C	D	E	F	G	Z
h(node)	8	4	6	6	3	6	2	0

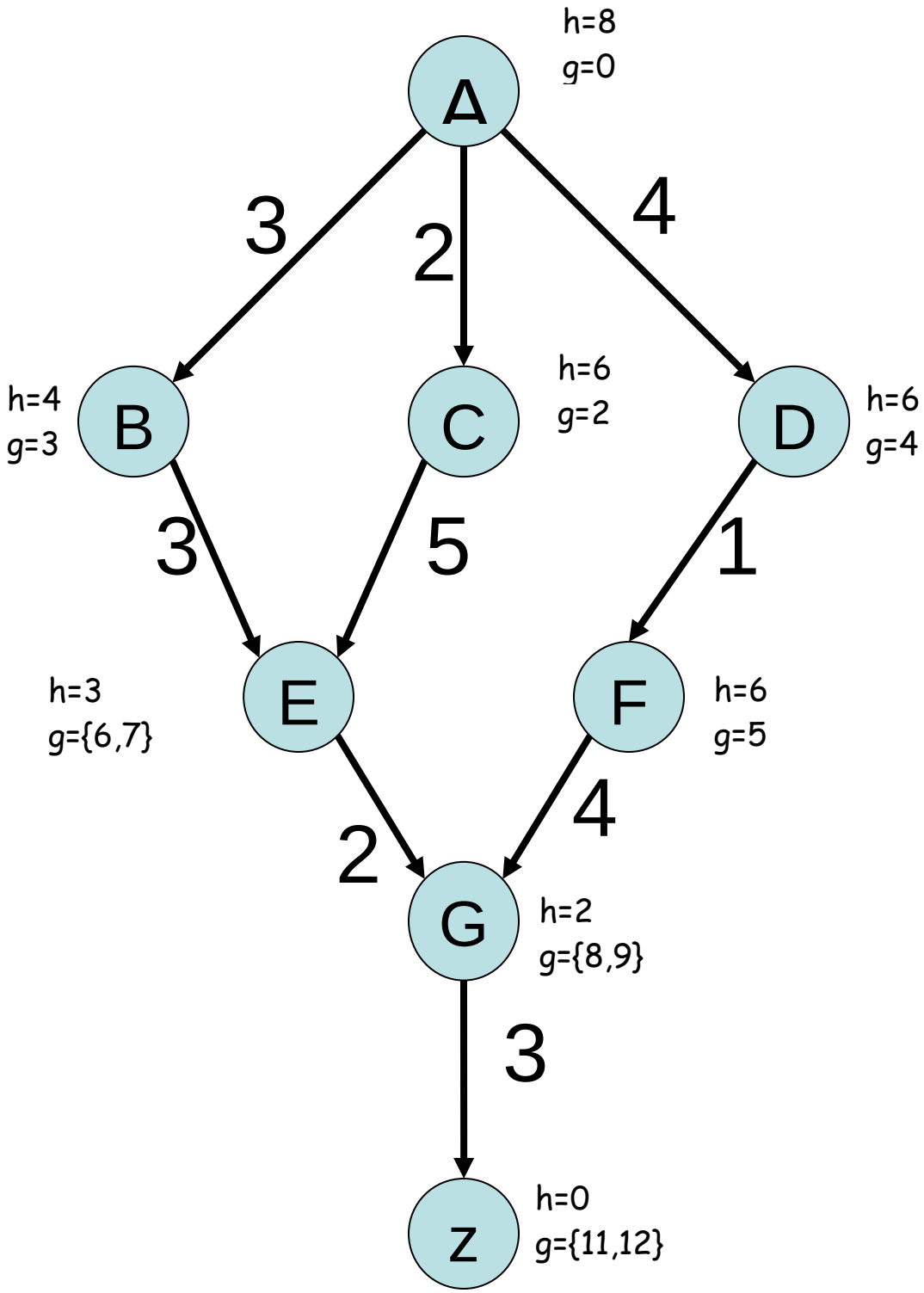
There are several places where nodes are tied for being the best to expand next. Here are all of the possible sequences

node	A	B	C	D	E	F	G	Z
order	**1**	**2**	**3**	**6**	**4**	**6**	**5**	**7**

node	A	B	C	D	E	F	G	Z
order	**1**	**2**	**3**	**6**	**4**		**5**	**6**

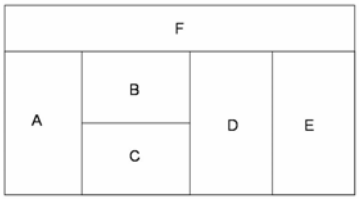
node	A	B	C	D	E	F	G	Z
order	**1**	**2**	**3**	**5**	**4**		**6**	**7**

node	A	B	C	D	E	F	G	Z
order	**1**	**2**	**3**	**5**	**4**	**7**	**6**	**8**



The h and g values need not be written on the graph, but make it easier to understand the state space and to figure out how graph search will proceed.

7. Constraint Satisfaction (30 points)



In the map-coloring problem shown to the right, each variable (A,B,C,D,E, and F) has the domain {r,g,b} To solve this problem, we use the simple backtracking algorithm presented in class and in the text with the **forward-checking** operation. No other constraint-propagation operation (e.g., AC3) is used. Recall that when a value is assigned to a variable V, forward checking eliminates inconsistent values in the variables adjacent to V in the constraint graph. Use these heuristics, in this order, to select the next variable to assign: (i) most-constrained-variable (aka, minimum remaining values), (ii) most-constraining-variable (aka, degree heuristic), (iii) least-constraining-value. When several variables tie, select the first in alphabetic order. When several values tie, select them in the following order: r, g, b.

a) [5] Draw the constraint graph for this problem: a graph with a node for each variable and an arc between two variables if there is a constraint between them.
b) [5] Which variable will be selected first by the algorithm? Why?
c) [5] Which value will be assigned to this variable? Why? Which values of which variable domains does the forward-checking operation then remove?
d) [5] Which variable will be selected next? Why?
e) [5] Which value will be assigned to this variable? Why? Which values of which variable domains does the forward-checking operation then remove?
f) [5] Use your answers to the above questions to count the number of complete valid assignments for this problem. Tell us how you proceed to do the counting.

1. The most-constrained-variable heuristic does not give any preference among the variables. But the most-constraining-variable heuristic is applicable, and gives B, D, and F as the most promising variables to instantiate. So, B is selected since it comes first in alphabetic order.

2. The least-constraining-value heuristic does not give any preference among the possible values {r,g,b} of B. So, "r" is selected. Forward checking then removes the value "r" from the domains of A, C, D, and F.

3.The most-constrained-variable heuristic gives the same preference to A, C, D, and F that all have two remaining values in their respective domains. The most-constraining-variable heuristic gives preference to variables D and F that are both involved in 3 constraints involving variables not yet instantiated (in contrast, A and C are only involved in two such constraints). D is selected.

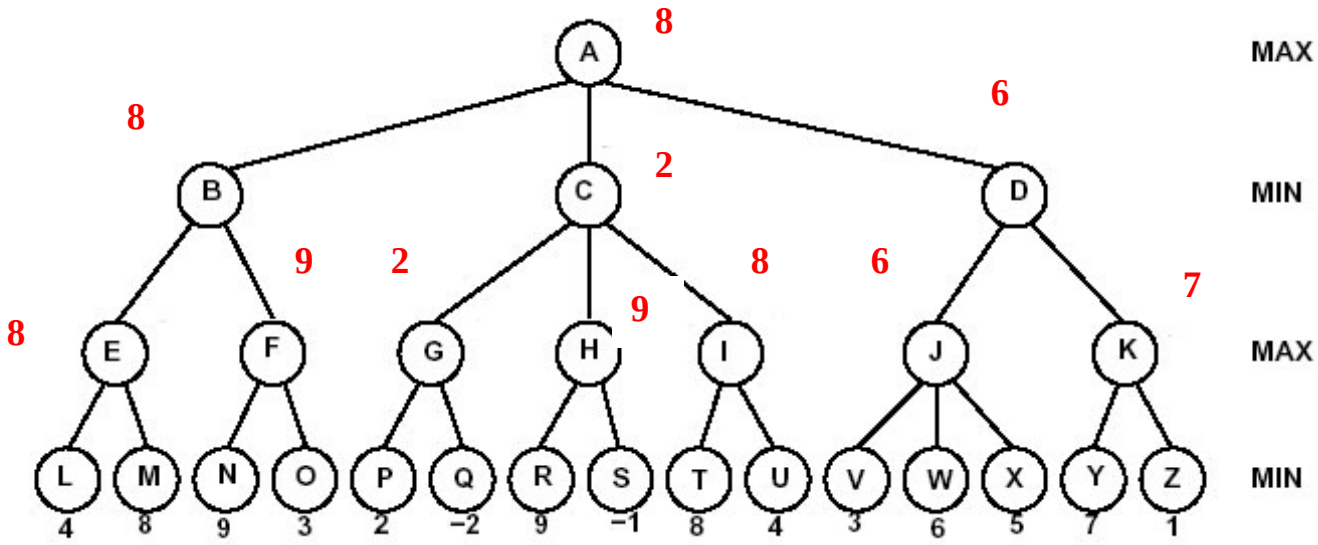
4. The least-constraining-value heuristic gives no preference among the possible values {g, b} of D. So, "g" is selected. Forward checking then removes the value "g" from the domains of C, E, and F.

5. Both F and C now have the reduced domain {b}, so by necessity their value must be "b". Assigning this value to both variables leads forward checking to reduce the domain of A to {g}, hence to give the value "g" to A. The domain of E is reduced to {r}, so E is assigned the value "r".

Initially, 3 values were possible for B, then 2 for D. There was no other choice of values. So, in total there are 3×2 = 6 complete valid assignments.

8. Game Trees (30 pts)

Consider the following game tree. Assume it is the maximizing player's turn to move. The values at the leaves are the static evaluation function values of the states at each of those nodes.



a) [10] Label each non-leaf node with its minimax value. **See above**

b) [5] Which move would be selected by Max? **B**

c) [10] List the nodes that the alpha-beta algorithm would prune (i.e., not visit). Assume children of a node are visited left-to-right. **O H R S I T U K Y Z**

d) [5] In general (i.e., not just for the tree shown above), if we traverse a game tree by visiting children in right-to-left order instead of left-to-right, can this result in a change to (i) the minimax value computed at the root? (ii) The number of nodes pruned by the alpha-beta algorithm? **(i) no, (ii) yes**