Name ____________________

ID # _____________________

CS 520 - Midterm Exam

March 13, 2002

1. [15 Points] State Spaces

Consider the problem of finding a path through a maze. The maze is constructed on a grid of squares. You may step from a square to its neighbors up, down, left, and right, but not diagonally. Some squares are black, some white; you may not step in a black square.

A problem is specified as an array of bits representing the colors of the squares, an (x, y) pair giving the start position and an (x, y) pair giving an end position. An answer is a list of (x, y) pairs giving a legal path from the start position to the end position. It is important to find the shortest path.

Consider this as a state space search problem.

A. How would you represent a state? **A state is a square, ie xy pair**

B. Describe the operators **Up, down, left right: move to square one step up (or down, etc) from current. Preconditions: new square not black**

Describe the start and end states **start is the postion specified in the problem, similar for end**

C. If you had to choose between depth first and breadth-first search for this problem, which would you choose? Why? **breadth first since depth first may not find shortest path – in fact depth first might never termionate**

D. If you had to choose between breadth-first and A* which would you choose? Why? **A* is faster**

2. [20 Points] CSP

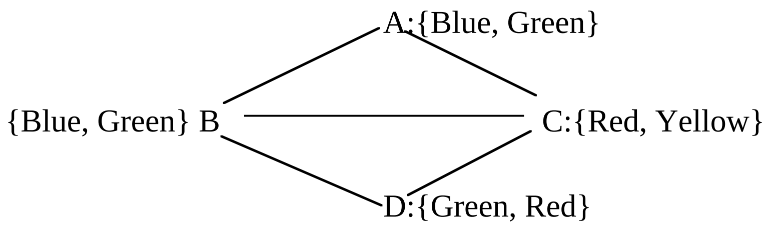
Consider the problem of assigning teaching assistants to recitation section as a constraint satisfaction problem. Suppose each course has one or more lecture sections and each lecture section has one or more recitation sections. A TA teaches two recitation sections;

both must be from the same lecture section. Each section has a time, and each TA has a set of times he or she is available forteaching. Suppose we already know which lecture section each TA is assigned to, but we need to choose the specific sections.

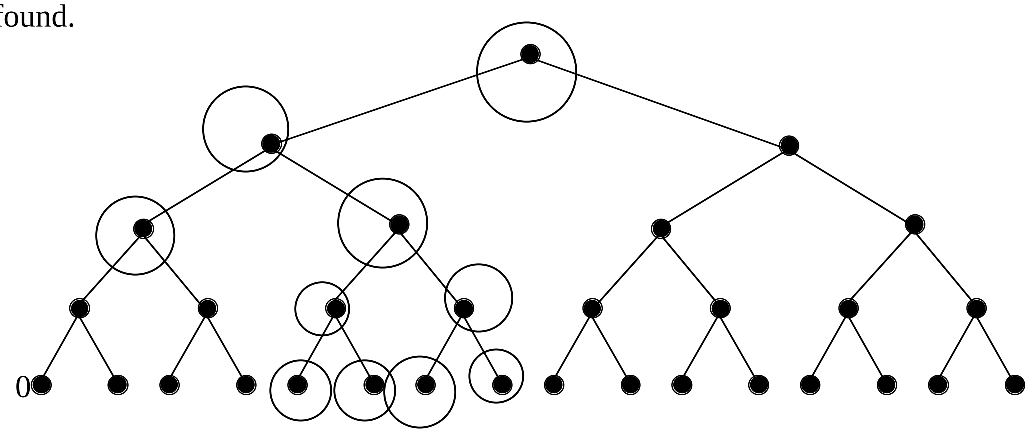
A. Suppose the variables are the sections. What is the domain of a variable? **theTA's assigned to the corrseponding lecture**

B. Suppose the variables are the sections. What are the constraints? **If two sections share a TA they must meet at different times; at most 2 sections share a TA, if a TA is the value of a section, TS's and section's times must match**

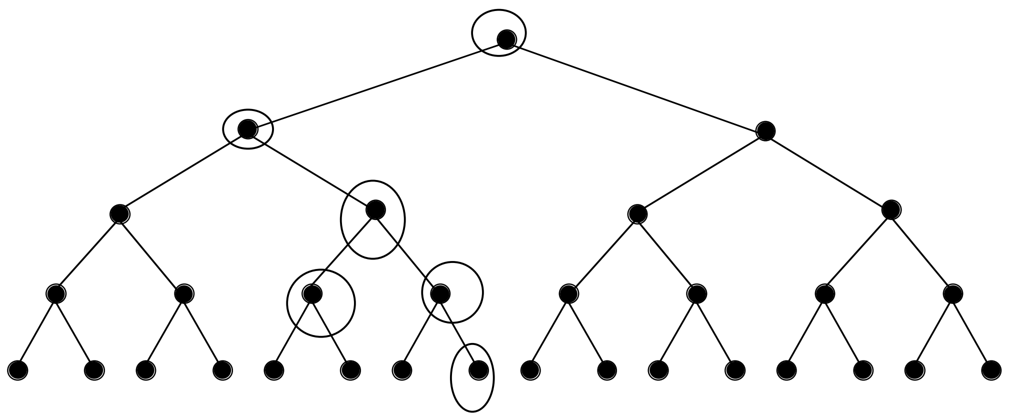
For the questions below, assume we are doing graph coloring (nodes connected by an arc must have different colors) on the graph below. The possible colors for each node are listed beside the node. Assume we process both variables and values in alphabetical order – e.g. node A before B and Blue before Green.



C. On the tree of partial assignments below, circle the nodes we will reach if we are doing early pruning but no forward checking. Assume we stop as soon as one solution is found.

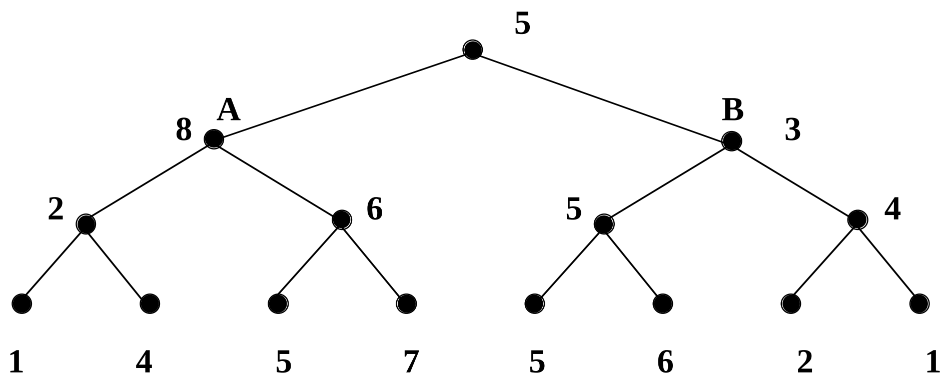


D. On the tree of partial assignments below, circle the nodes we will reach if we are doin0g forward checking. Assume we stop as soon as one solution is found.



3. [20 Points] Games

In the game tree below, each node has a number next to it that gives the value returned by the static evaluator on that game state. The player who moves first wants to maximize the values.



E. Suppose we look ahead one level (i.e. down to the nodes labeled A and B) and then back up the values.

What value will we give to the root node? ___8____

What move will we make? Circle one of the letters here: [✓ A] B

F. Suppose we look ahead two levels and then back up the values.

What value will we give to the root node? __ 4_____

What move will we make? Circle one of the letters here: A [✓ B]

G. Suppose we had a game in which players did not win or lose, but were rewarded to a greater or lesser extent. E.g., one outcome might give player A $10 and player B $5, while another outcome might give player A $8 and player B $20. What changes would we have to make to the minimax algorithm for backing up game-state values?
We need to keep two scores per node: values to player1 and 2.
To back up scores to a node n at level i, where it is player p's turn to move, we choose the child whose score for player i is largest from all the children of n, and back up its values for both players to n.

4. [20 Points] Lisp

A. Suppose we are writing a tic-tac-toe program as in the homework. A board is represented by a 3x3 array. Finish the function mix-count on the next page. The code that was handed out for scan-fn and scan-row are below, along with brief descriptions of a few lisp functions Your solution must use function list-scan appropriately. You may also use scan-row, scan-column, and scan-diagonal. You may not use all-lines.

```
;;;for each square in specified row, call function with 3 args:
;;;   contents of square, row-number column-number
(defun scan-row (board row function)
  (dotimes (column 3)
    (funcall function (aref board row column) row column)))

;;; scan-fn is a function like scan-row.  arg specifies row or
;;; equivalent, i.e. 2nd arg to scan-fn.   Returns a list of
;;; the contents of the squares scan-fn looks at
(defun list-scan (scan-fn board arg)
  (let ((result nil))
    (funcall scan-fn board arg
          #'(lambda (mark row column)  (push mark result)))
        result))
```

Lisp:
(**let** ((var1 expr1)(var2 expr2)…) expra exprb …) creates a local scope in which the vars are bound to expr1, 2, etc, and evaluates expra, b, etc in this scope, returning the value of the last one.
(**do-list** (var expr1) expra exprb …) evaluates expr1 to get a list, then binds var successively to each element of the list and evaluates expra, b, … with each binding.
(**incf** var) is like (setq var (+ 1 var)), i.e. like var++ in c.
(**list** expr expr …) evaluates the exprs and returns a list of the values.
(**member** string list :test #’string=) returns true if string is an element of list, false otherwise
(**and** …) and (**or** …) do just what you would expect.
#’sym returns the function meaning of symbol sym

```
    ;;; Counts the number of rows, columns, and diagonals that have
    ;;; BOTH an “x” and an “O” in them.  E.g., for
    ;;;  X | O | O     the result is 3 (the top row, middle column,
    ;;; -----------    and a diagonal)
    ;;;    | X |
    ;;; -----------
    ;;;    |   |
    (defun mix-count (board)
      (let ((lines nil))
        (dotimes (r 3)
          (setq lines (cons
                       (list-scan #'scan-row board r)
                       lines)))
        (dotimes (c 3)
          (setq lines (cons
                       (list-scan #'scan-column board c)
                       lines)))
        (dotimes (d 2)
          (setq lines (cons
                       (list-scan #'scan-diagonal board d)
                       lines)))
        (let ((count 0))
          (dolist (l lines)
            (if (and (member "X" l :test #'string=)
                     (member "O" l :test #'string=))
                (incf count)))
          count)))
```

5. [10 Points] English to FOL

Express each of the following in First Order Logic or explain why it cannot be done.

A. Every person who likes ice cream also likes custard.
∀p (person(p) and likes(p , icecream)=> likes(p, custard)

B. If there is any food a person doesn't like, then that person doesn't like Brussels Sprouts.
∀p person(p) => ((∃ f food(f) ^ ~likes(p, f)) => ~likes(p, brusselssprouts)))

C. If it is raining then everyone who does not have an umbrella is getting wet.
|raining =>(∀ x ~have(x, umbrella) => getting-wet(x))

D. You must not pass a school bus when its red lights are blinking.
Cannot be done – "must" is not expressible in FOL

6. [15 Points] Situation Calculus

Suppose we have a game like Wumpus. Suppose the agent can do the following actions:

Pickup(x):	If x is portable and in the same room as the agent, this results in the agent holding x.
Putdown(x):	If the agent is holding x this results in the agent not holding x.
GoNorth:	The agent and everything it is holding move to room one room north of the current room

Suppose we use the following predicates:

Holding(x, s): the agent is holding object in state s
At(x, room, s): x is the agent or an object. X is in the specified room in state s
Portable(x): x can be carried. Northof(room1, room2) room1 is one room north of room2

Write the axioms that reflect the effect of the actions on the predicate At
at(x, room,resultof(GoNorth, s)) ⇔(x=agent v holding(x, s)) ^ at(x, room1, s) ^ Northof(room, room1)
at(x, room,resultof(act, s))<= (act = pickup v act = putdown)^at(x, room,s)

7. [15 Points] Inference

Given the following knowledge base:

Larger(elephant, lion)
Larger(lion, fox)
Larger(X, Y) ^ larger(Y, W) => larger(X, W)
Eats(X, Y) => larger(X, Y)
Eats (fox, mouse)

Write out a proof using backchaining generalized modus ponens of Larger(elephant, mouse). For each unification, show the resulting substitution.

```
larger(elephant, mouse)
unifies with larger(X,W).  substitution = X/elephant, W/mouse
  larger(elephant,Y)
   unifies with larger(elephant, lion):  Y/lion
  larger(lion,mouse)
    unifies with  larger(X, W):  X /lion,W/mouse
      larger(lion,Y)
        unifies with larger(lion, fox):  Y/fox
     larger(fox,mouse)
        unifies with larger(X, Y): X/fox,Y/mouse
        eats(fox,mouse)
        unifies with eats(fox,mouse)
```