Final Exam
CS 520: Introduction to Artificial Intelligence
Fall, 1999

Name _____

ID _____

For grader's use:

1 _____/ 40

2 _____/ 15

3 _____/ 15

4 _____/ 20

5 _____/ 30

6 _____/ 40

7 _____/ 20

8 _____/ 20

total _____/ 200

1. [**40 points**] For each of the problems below, say which of these approaches is best used for the problem and why.

   - Constraint Satisfaction
   - State Space Search
   - Planning
   - Expert Systems
   - None of the Above

   Problems:

   (a) Missionaries and Cannibals

   State-space search: it does not have complex states so planing is not appropriate and it does not fit the structure of constraint satisfaction, nor is it an expert problem.

   (b) N-Queens

   This does have the structure of a constraint satisfaction problem, so that is the best to use.

   (c) Choosing a set of airplane flights for a vacation

   This could either be constraint satisfaction or search, depending on the structure of your criteria for an acceptable set of flights/

   (d) Choosing a new car to buy

   Expert system, since it involves figuring out exactly what the criteria for choosing a car should be.

2. [**15 points**]

   Finish the lisp function "everyother" below. You may assume the argument is non-empty list. everyother returns a new list containing every other element of the argument, starting from the first element. E.g.,

```
(everyother '(a b c d) --> (a c)
(everyother '(a b c d e) --> (a c e)
(everyother '(a b) --> (a)
(everyother '(a) --> (a)
```

   You may not use the functions nth or elt. You may not use iteration (e.g., do, dolist, while, or loop). You must use recursion.

```
(defun every-other (list)
  (if (null (cdr list))
    list
  (cons (car list)(every-other (cdr (cdr list))))))
```

```
)
```

3. [**15 points**] Write everyother in prolog, e.g.,

```
:-everyother([a, b, c], X).   X= [a, c]
```

```
everyother([A],[A]).
everyother({A, B |Rest], [A | ER]):-everyother(Rest,ER).
```

4. [**20 points**] Suppose that you wanted to describe the task of writing a program as a planning problem. Assignment statements of the form

   ```
   <variable1> =  <variable2>;
   ```

   might represented by the following operator:

   ```
   assign-val(Var1, Val2):
     Preconditions:  holds(Var1, Val1), holds(Var2, Val2)
     Postcondition:  holds(Var1, Val2), not holds(Var1, Val1)
   ```

   Suppose you have the following planning problem:

   - Initial State: holds(v1, x), holds(v2, y)
   - Goal State: holds(v1, y), holds(v2, x)

   You may assume that in addition to variables v1 and v2, another variable, v3, is also available for use.

   Draw a partial-order plan that solves this problem using only the operator assign-val. (Of course you will need to use it more than once.) Be sure to make it clear in your diagram which arcs are of which types.

   see http://www.cs.rutgers.edu/ lou/520-plan.gif

5. **[30 points]**

Suppose a class Introduction to Computer Science is full, and we decide to let in only those students likely to pass the course. Suppose we have data about the students from last semester that lists

- their major (one of cs, math, or "other")
- their class year (senior, juniors, and so on)
- their grade in calculus (A, B, or C)
- their grade in Introduction to Computer Science (Pass or Fail)

(a) For this problem, if we wanted to learn with a backpropagation neural network, what would be the input signals and the output signals?

inputs: 2 inputs encoding each of the following: major, class, grade, plus one input for grade

(b) Can a perceptron network learn an exact function for the following data? Why or why not?

No, because students c and f have identical attributes but one is an instance of the concept passing-student and one is not.

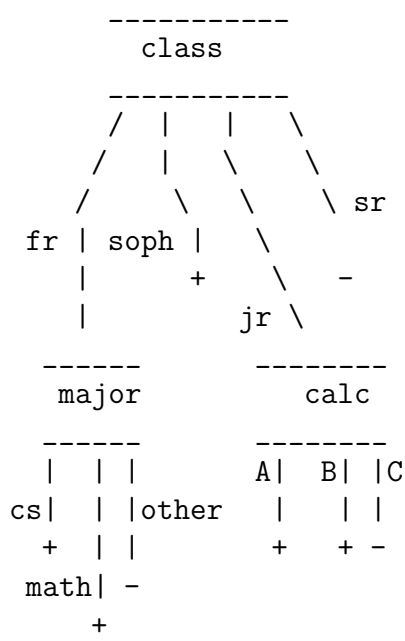
TABLE ONE

Student	Class	Major	Calc	CS grade
a	Jr	CS	A	Pass
b	Jr	Math	B	Pass
c	Jr	CS	B	Pass
d	Jr	CS	C	Pass
e	Sr	Math	A	Fail
f	Jr	CS	B	Fail
g	Jr	Math	C	Fail
h	Jr	Math	A	Fail

6. **[40 points]**

Suppose we use Decision Trees instead of neural nets to learn which students are likely to pass.

(a) Suppose we learned the following Decision Tree:

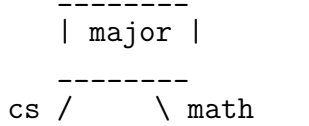


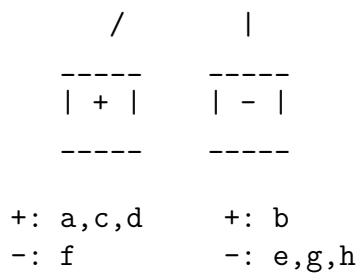
Does this tree give the correct prediction for the following student:
Major: cs, Class: Jr., Calc grade: B, CS grade: pass

YES is correct

(b) Write a statement in first order logic that is the equivalent of this tree.
$\forall s(pass - cs(s) <=>$
$freshman(s) \vee (csmajor(s) \wedge mathmajor(s))$
$\vee\ soph(s)$
$\vee\ jr(s) \wedge (calcA(s) \vee calcB(s)$

(c) Suppose we had the data in Table One (on the previous page) and wanted to build a decision tree. Show the result after one iteration, that is, after only the root node has been put into the tree.




/
|
+
-
+: a,c,d
-: f
+: b
-: e,g,h

7. [**20 points**] Suppose we wanted to solve the following problem using a Genetic Algorithm. Genetic algorithms will not be on the exam

8. **[20 points]** Suppose there was an computer game in which the player shoots a gun at a target that looks like this

   ```
   X O X
   ```

   To win you must hit each of the X's at least once and NOT hit the O. You have as many shots as you like, but you lose one point for each shot. If you win you get 100 points and if you hit the O you lose 100 points. You can choose to aim the gun at any letter. If you aim at a letter there is a 80% chance of hitting it, a 5% chance of hitting each of the other letters, and a 10% chance of hitting nothing.

   Suppose we want to represent this as a reinforcement learning problem.

   (a) Draw the state space. (Hint: all that matters about a letter is whether it has been hit yet or not.) Label your states appropriately. Do NOT show transition probabilities in this picture.

   See http//www.cs.rutgers/ lou/520/old-final-states.gif

   (b) What are the actions?

   aim at left x, aim at right x aim at o

   (c) What are the rewards?

   state left X hit / right X hit / o not hit: +100 states with o hit: -100 other states: -1

   (d) Choose any one state and any one action and write out the corresponding transition matrix M.

   State: nothing hit
   Action: aim at left x

next state			Probability
left-x	o	right-x	
hit	hit	hit	0
hit	hit	not-hit	0
hit	not-hit	hit	0
hit	not-hit	not-hit	.8
not-hit	hit	hit	0
not-hit	hit	not-hit	.05
not-hit	not-hit	hit	.05
not-hit	not-hit	not-hit	.1