CS 540-1: Introduction to Artificial Intelligence

Exam 2: 11am-12:15pm, November 21, 1994

CLOSED BOOK

(one page of notes allowed)

Write your answers on these pages and show your work. If you feel that a question is not fully specified, state any assumptions you need to make in order to solve the problem. You may use the backs of these sheets for scratch work.

Write your name on this and all other pages of this exam. Make sure your exam contains six problems on six pages.

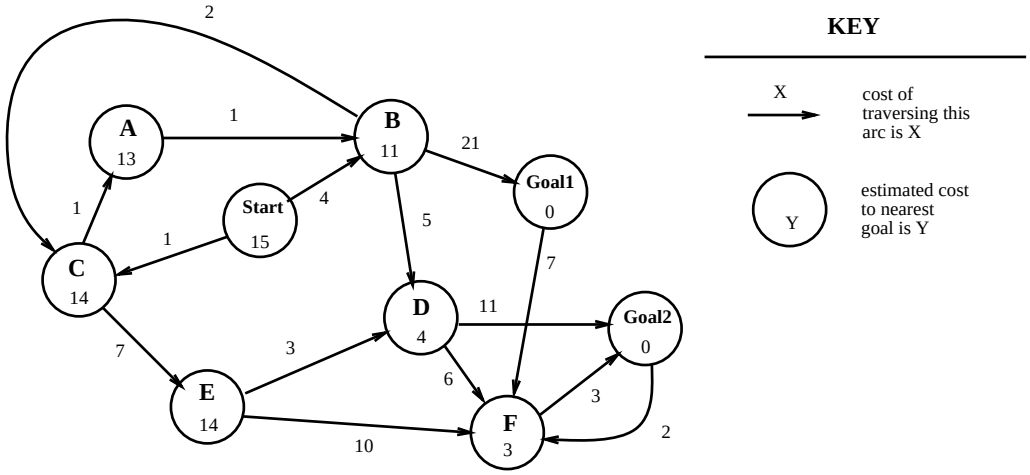
Name _____

Student ID _____

Problem	Score	Max Score
1	_____	30
2	_____	22
3	_____	18
4	_____	12
5	_____	6
6	_____	12
Total	_____	100

PROBLEM 1 - Search (30 points)

Consider the search graph drawn below; the start and goal states are labeled. *Note that arcs are directed.* For each of the search strategies listed below, indicate which goal state is reached (if any) and list, in order, the states expanded. (A state is *expanded* when it is *removed* from the OPEN list.) *When all else is equal, nodes should be expanded in alphabetical order.*



Depth-First Search

Goal state reached: _____ States expanded: _____

Iterative Deepening

Goal state reached: _____ States expanded: _____

Best-First Search (using $f=g+h$)

Goal state reached: _____ States expanded: _____

Hill Climbing (using the h function only)

Goal state reached: _____ States expanded: _____

Beam Search (with a beam width of 2)

Goal state reached: _____ States expanded: _____

Would the h function in this graph lead to an *admissible search*? _____

Explain your answer:

PROBLEM 2 - Forward-Chaining: Production Systems (22 points)

(a) Consider the following rule base for a production system:

```
(1)  p(X) ∧ q(Y) ∧ r(X,Y)  →  assert s(X)
(2)  p(X) ∧ q(X) ∧ s(X)    →  retract s(X)
(3)  r(X,Y)  →  assert s(X) ∧ retract p(X) ∧ retract q(Y)
```

Assume working memory (WM) is currently of the form:

```
p(1) ∧ p(2) ∧ q(1) ∧ q(2) ∧ r(1,2) ∧ r(2,1) ∧ r(2,2)
```

Finally, assume that the conflict-resolution scheme used is the same one used in HW5 (i.e., pick the top-most rule whose preconditions are satisfied; however, a given rule cannot be used twice with the same bindings).

Show below the first five steps this production system would perform (even though you only need to find the top-most acceptable rule, you must still show the *full* conflict set):

Cycle	*Conflict Set*	*Rule Fired*	*Changes to WM*
1			
2			
3			
4			
5			

(b) Imagine you wish to create an ‘‘email agent’’ by writing a production system that preprocesses your incoming email. For simplicity, assume someone gives you a program that parses your message into an FOPC representation along the lines of:

```
from(smith) ∧ to(jones) ∧ to(lee) ∧ cc(starr) ∧ subject("today's meeting") ∧ ...
```

Write two useful production rules for this task. Explain what these rules do. (The two rules should have reasonably different preconditions and effects.)

(1)

(2)

PROBLEM 3 - Backward-Chaining: Prolog (18 points)

a) Consider the following Prolog rulebase:

```
p(1,2).
p(3,3).
p(2,1).
q(1,2).
q(2,1).
q(2,2).
q(3,3).
r(2,1).
r(2,2).
s(X1,Y1)   :-  p(X1,Y1), q(Y1,X1), r(X1,X1).
s(X2,Y2).
r(X3,Y3)   :-  q(X3,Y3).
w(X4,Y4)   :-  p(X4,Y4), q(2,Z4), r(X4,Z4).
```

For the following queries, report the answers that Prolog a requested answer. Report *fail* is Prolog would not be able to find a requested answer.

i) `?- s(2,1).` First Binding List Found: _____

Second Binding List (if one): _____

Third Binding List (if one): _____

ii) `?- w(U,V).` First Binding List Found: _____

Second Binding List (if one): _____

Third Binding List (if one): _____

b) Write a Prolog program for *sisters(X,Y)*, which determines if *X* and *Y* are sisters. You may only assume that the predicates *grand_parent(X,Y)*, *parent(X,Y)*, *ancestor(X,Y)*, *male(X)*, *female(X)*, *tall(X)*, *young(X)*, *older_than(X,Y)*, and *different(X,Y)* are defined. *Different(X,Y)* is true when *X* and *Y* have been bound to different constants; the semantics of the other predicates should be obvious (the binary ones, *predicate(X,Y)*, can be read ‘‘*X* is a *predicate* of *Y*’’).

PROBLEM 4 - Important AI Concepts (12 points)

In the space below, provide *brief and succinct* summaries of the ***importance*** of the following AI concepts.

---

horizon effect

---

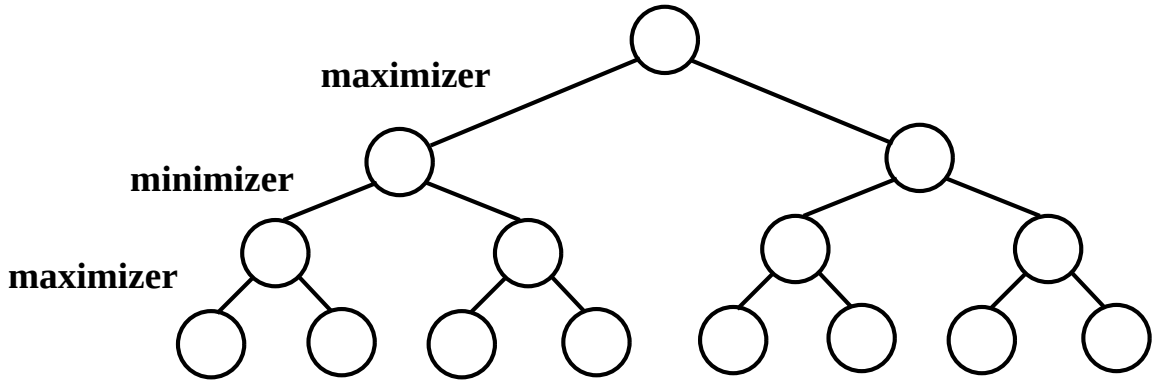
interacting subgoals

---

negation-by-failure

PROBLEM 5 - Game Playing (6 points)

Assume that a game has a static-board evaluator (SBE) that only returns one of three values: -1 (lose), 0 (draw), and 1 (win). Consider the game tree drawn below. What is the *fewest* possible number of calls to the SBE? To illustrate your answer, label those leaf nodes that would have to be called with the score that the SBE would have to return. (If you feel there are several way that the fewest calls could occur, you need only show *one* of them.)



PROBLEM 6 - Lisp (12 points)

The function defined below, `(mark-doubles X Y)`, is supposed to replace, by the atom *twice*, all those sexpr's in list *X* that are also in the *top-level* of list *Y*. That is, it should operate as follows:

```
> (mark-doubles '(1 2 3) '(a b c))
(1 2 3)

> (mark-doubles '(1 (2) 3) '(3 this list sure is long (1) 2))
(1 (twice) twice)

> (mark-doubles '(1 (hi there) 3) '(hi there))
(1 (twice twice) 3)

> (mark-doubles '(1 (hi there) 3) '(an aloha (hi there) to you))
(1 twice 3)
```

```
(defun mark-doubles (X Y)

 "This code is buggy."

 (cond ((member X Y)

        twice)

       ((atom X)

        nil)

       (t (dolist (item X)

            (mark-doubles item Y))

          )
       )
 )
```

Your task is to debug this function. To the *right* of the code, describe the errors in it and show the necessary corrections.