CS 540-1 -- Introduction to Artificial Intelligence

Exam 1 - October 2, 1992

CLOSED BOOK

90 minutes

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 seven problems on six pages.

Name ____

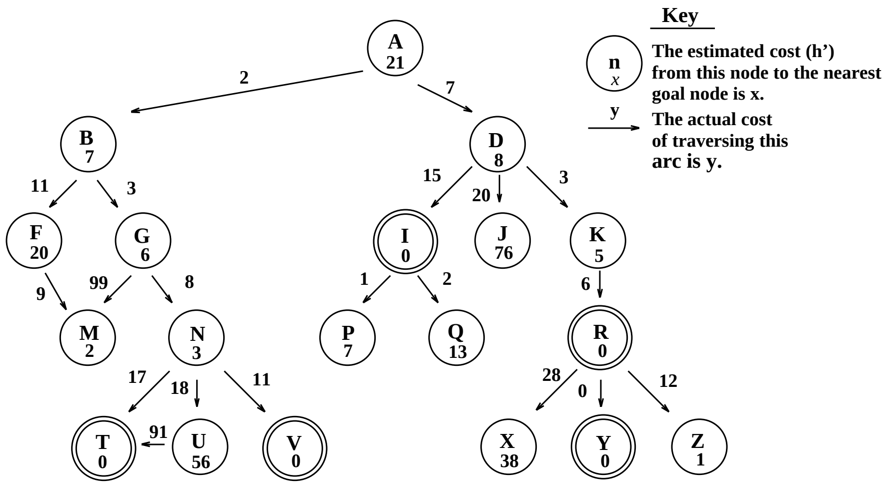
Student ID ____

Problem	**Score**	**Max Score**
1	____	30
2	____	15
3	____	15
4	____	10
5	____	10
6	____	10
7	____	10
Total	____	100

(over)

PROBLEM 1 - Search Strategies (30 points)

Consider the search graph drawn below. The initial state is at the top, and goal states are represented by double circles. *Note that arcs are directed.* Which goal state is reached will depend on the search strategy applied. For each of the search strategies listed below, indicate which goal state is reached (if any) and list, in order, the states explored. (A state is *explored* when the item containing it is *removed* from the OPEN list.) Assume that the NEXT-STATES function returns a state's successors in the same left-to-right order as in the search graph.


Key
n
x
The estimated cost (h') from this node to the nearest goal node is x.
y
The actual cost of traversing this arc is y.
A 21
2
7
B 7
D 8
11
3
15
20
3
F 20
G 6
I 0
J 76
K 5
9
99
8
1
2
6
M 2
N 3
P 7
Q 13
R 0
17
18
11
28
0
12
T 0
91
U 56
V 0
X 38
Y 0
Z 1

Depth-First Search

Goal state reached: _____ States explored: ____

Breadth-First Search

Goal state reached: _____ States explored: ____

Best-First Search

Goal state reached: _____ States explored: ____

Beam Search (with a beam width of 2)

Goal state reached: _____ States explored: ____

Hill Climbing

Goal state reached: _____ States explored: ____

A* Search

Goal state reached: _____ States explored: ____

PROBLEM 2 - Optimal Solutions and Heuristic Functions (15 points)

The search algorithm A^* produces optimal solutions provided the heuristic estimate, *h'*, never overestimates the actual least cost, *h*.

Draw a *simple* search space where *h'* does overestimate *h* and the A^* algorithm returns a *non*-optimal solution. Explain your answer.

PROBLEM 3 - Short Questions on Search (15 points)

a) Under what circumstances would it make sense to go "down hill" (ie, explore a child whose heuristic value is worse than the parent's) when executing the hill-climbing search algorithm? When would it *not* make sense?

b) When is breadth-first search an *admissible* search strategy? Briefly explain.

c) Name two (2) searches that may never find a solution, even when one exists. Briefly explain.

PROBLEM 4 - Executing Lisp (10 points)

Assume the following is typed to a newly-started LISP:

```
(setf a (+ 3 5))
(setf b ’(8 (9)))
(setf c ’(cons b c))
(setf d ’(8 (9)))
(setf e d)
```

What does each of the following return (write "error" if an error condition develops):

a) `a`

value = ____

b) `(- 31 a)`

value = ____

c) `(first (first (rest b)))`

value = ____

d) `(cons b c)`

value = ____

e) `(list b c)`

value = ____

f) `(append b c)`

value = ____

g) `(list ’a a)`

value = ____

h) `(equal b d)`

value = ____

i) `(eq d e)`

value = ____

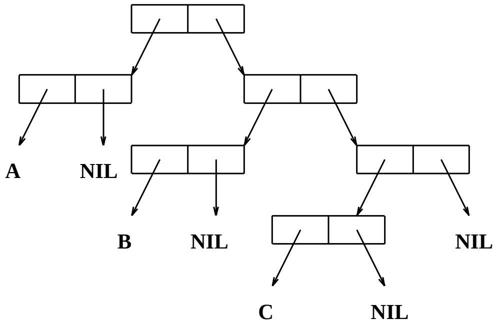
j) `(eq b e)`

value = ____

PROBLEM 5 - Cons-Cell Notation (10 points)

i) Represent the list **(1 (2 (3)) 4)** in *cons-cell notation*.

ii) Convert the following to *list notation*.



ANSWER: ____

PROBLEM 6 - Understanding Recursive Functions (10 points)

Consider the following recursive function definition:

```
(defun F (x)
  "This is a mystery function."
  (if (atom x)
       0
       (if (numberp (first x))
             (+    (first x)  (F (rest x)))
             (+ (F (first x)) (F (rest x))))))
```

a) What is returned by the following:

i) `(F '(7 is prime and 9 is not))`

value = ____________________

ii) `(F '(a (b 13) ((45) 1)))`

value = ___________________

b) Briefly describe what this function computes (don't merely paraphrase the Lisp code).

PROBLEM 7 - Writing Recursive Functions (10 points)

Consider the function `subset`, which takes two arguments, `set1` and `set2`, both represented as lists of atoms. It returns non-nil only when `set1` is a subset of `set2` (i.e., every member of `set1` is also a member of `set2)`.

Write a *recursive* version of `subset` in Lisp.