(PRINT) Name _____Student No_____

Signature _____Total Mark_____/100

The University of Toronto

Computer Science 384 – Introduction to Artificial Intelligence

Midterm Test 2
2003 March 21

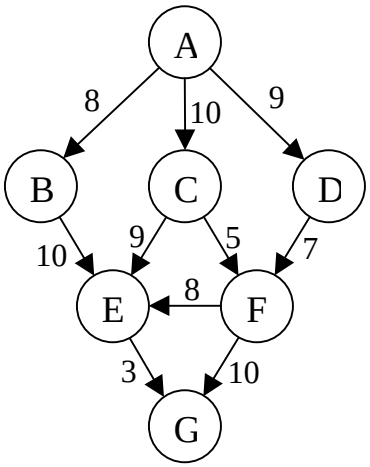
Time: 45 minutes
Total marks: 100

Answer all questions on this paper. No books or other materials may be used. Calculators and personal computers are not permitted.

This examination has 7 pages. Check that you have a complete paper.

1	
2	
3	
Total	

1) [32 pts] Consider the following graph and 3 heuristics for estimating the cost of reaching the goal node G from any other node:



Node	h1	h2
A	6	21
B	3	13
C	1	3
D	2	17
E	1	1
F	4	12
G	0	0

a) [12 pts] Indicate for each heuristic, h1 and h2, whether a least cost path is guaranteed to be found when searching for a path from any node to the goal node G using A*? Explain briefly why.

h1: yes [3 pts].
h1 is admissible (h(n, g) ≤ d(n, g)) so least cost path guaranteed [3 pts].

h2: no [3 pts].
h2 is not admissible (h(n, g) > d(n, g)) so least cost path not guaranteed [3 pts].

b) [10 pts] Using heuristic h2 and A* search, indicate the path found from A to G and the number of nodes expanded.

Path: A B E G

Expanded 6 nodes (A B C E E G).

c) [10 pts] Using heuristic h2 and A* search with multiple path checking, indicate the path found from A to G and the number of nodes expanded.

Path: A C E G

Expanded 5 nodes (A B C E G).

2) [36 pts] Consider the following simple planning problem. You have lost your keys to open the lab door. Fortunately, a friend of yours is willing to lend you his keys but he is in Ottawa. Since you don't own a car you must rent a car to go to back and forth between Toronto and Ottawa. Furthermore, you have to make sure the fuel tank is full before each trip between Toronto and Ottawa. You are now about to plan your trip to Ottawa to borrow the keys and return them.

We describe the domain with the following propositions:

`haveCar`: you have a car
`tankFull`: the car tank is full
`haveKeys`: you have the keys to open the door
`doorOpen`: the door is open

We have five actions: `rentCar, getKeys, openDoor, returnKeys` and `refuel`. We describe these using the STRIPS representation allowing negative preconditions as follows. Note that some actions have empty precondition, add or delete lists.

Action	**Precondition**	**AddList**	**DeleteList**
`rentCar`	`neg(haveCar)`	`haveCar` `tankFull`	
`getKeys`	`neg(haveKeys)` `tankFull`	`haveKeys`	`tankFull`
`openDoor`	`neg(doorOpen)` `haveKeys`	`doorOpen`	
`returnKeys`	`tankFull` `haveKeys`		`tankFull` `haveKeys`
`refuel`	`haveCar`	`tankFull`	

See next two pages for the questions

a) [14 pts] Suppose we have the following start state represented using CWR: `[tankFull]`. Let the goal list be `[neg(haveKeys), doorOpen, tankFull]`. Show the steps a regression planner would go through to find a *shortest* plan for this problem. Choose only the action at each step that is in the plan and show the regressed sub goal list at each stage

Subgoals: [neg(haveKeys), doorOpen, tankFull]

Action: refuel

Subgoals: [neg(haveKeys), doorOpen, haveCar]

Action: returnKeys

Subgoals: [haveKeys, tankFull, doorOpen, haveCar]

Action: refuel

Subgoals: [haveKeys, doorOpen, haveCar]

Action: openDoor

Subgoals: [neg(doorOpen), haveKeys, haveCar]

Action: getKeys

Subgoals: [tankFull, neg(doorOpen), neg(haveKeys), haveCar]

Action: rentCar

Subgoals: [neg(doorOpen), neg(haveKeys), neg(haveCar)]

b) [10 pts] Suppose you decided to use a STRIPS planner with subgoal protection instead of a regression planner. STRIPS planners with subgoal protection can only find a plan when the goals are *serializable*. Is the goal list `[neg(haveKeys), doorOpen, tankFull]` serializable? If yes, give a goal ordering and explain why it is serializable. If no, explain why none of the goal orderings are serializable.

Serializable ordering: doorOpen, neg(haveKeys), tankFull [5 pts]

Once each goal is achieved it remains achieved in the plan listed in (a) [5 pts]

[5 pts] for wrong answer but consistent explanation.

c) [12 pts] Explain one advantage and one disadvantage of STRIPS planners (with subgoal protection) compared to regression planners when goals are serializable.

<u>Advantage</u>: Search with STRIPS may be faster/easier because of the "divide and conquer" approach that searches for one goal at a time [6 pts].

<u>Disadvantage</u>: Resulting plan may be longer (non-optimal) [6 pts].

3) [32 pts] Let's consider a reduced version of the "Sticks Game" where 4 sticks are arranged in the following configuration:

```
1st row:    |    (1 stick)
2nd row:  | | |  (3 sticks)
```

In this two-player game, players alternate to remove any number of sticks in a *single* row. The game ends when all the sticks have been removed and the loser is the player that removed the last stick.

Before answering this question, make sure to read all of a), b) and c). Answer a), b) and c) on the next page.

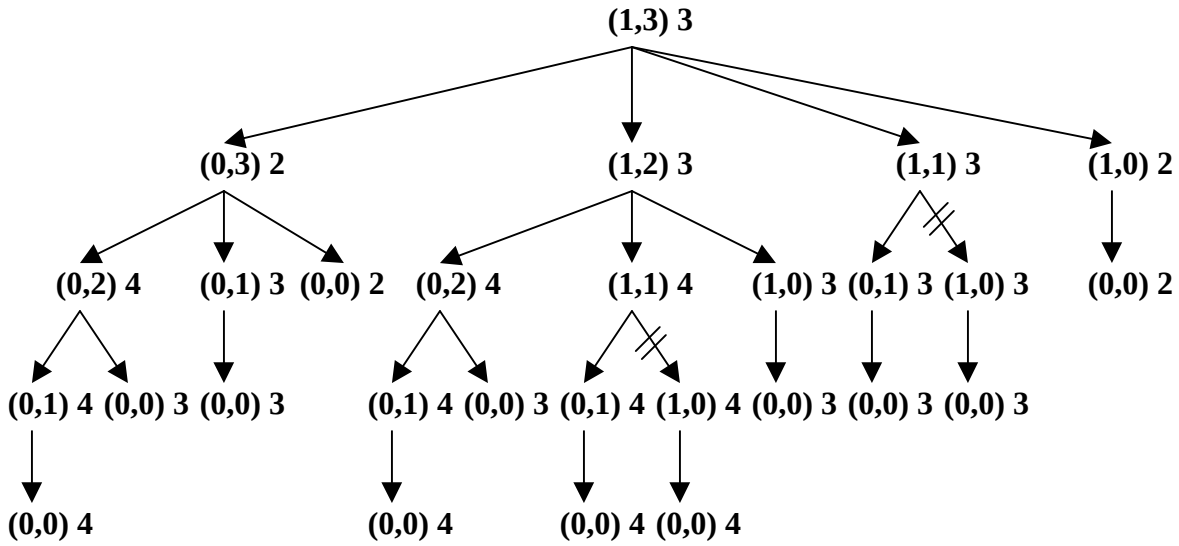
a) [12 pts] Show the game tree for the sticks game starting in the above configuration and where each node is labelled by a pair <N1, N2> indicating the number N1 of sticks in the first row and the number N2 of sticks in the second row. When expanding a node, list its children from left to right according to the following move ordering:

i) remove 1 stick from row 1
ii) remove 1 sticks from row 2
iii) remove 2 sticks from row 2
iv) remove 3 stick from row 2

The game tree gets pretty wide so please use *all* of next page to write it. Feel free to also write the tree sideways.

b) [10 pts] Suppose your computer uses this game tree to play, however you don't want it to play "perfectly", you'd like it to be a "challenging" opponent that will make the game last as long as possible (regardless of who wins). To do this, set the value of each leaf to its depth. Assume the root has depth 0 and that it is a max node. For each node of the tree, compute its minimax value and *clearly* write the value by *circling* it besides the node.

c) [10 pts] Suppose your computer uses DFS with alphabeta pruning to compute those minimax values. Indicate which parts of the game tree are pruned (i.e. never searched) assuming moves are considered in the order given above.



a) Tree

- **No pt deduction if order of branches or branch representation is different.**

b) Minimax

- **No pt deduction if tree is different.**
- **[-3 pts] if values at leaves are incorrect.**

c) Alpha-beta pruning

- **No pt deduction if tree is different**
- **[5 pts] for each cut.**
- **[-3 pts] for "extra" cuts.**