CS-271P, Intro to A.I., Winter Quarter, 2018—Quiz # 1—20 minutes

NAME:____

YOUR ID: ____________ ID TO RIGHT:___________ ROW:_____ NO. FROM RIGHT:______

1. (12 pts total, 3 pts each) TASK ENVIRONMENT. Your book defines a task environment as a set of four things, with the acronym PEAS. Fill in the blanks with the names of the PEAS

See Section 2.3.1.

Performance (measure) Environment Actuators Sensors

2. (48 pts total, 2 pts each) PROPERTIES OF TASK ENVIRONMENT. Your textbook (Fig. 2.6) gives several examples of task environments and their characteristics. Fill in the blanks with one of the underlined terms in the heading. The first one is done for you as an example.

See Fig. 2.6, Section 2.3.2.

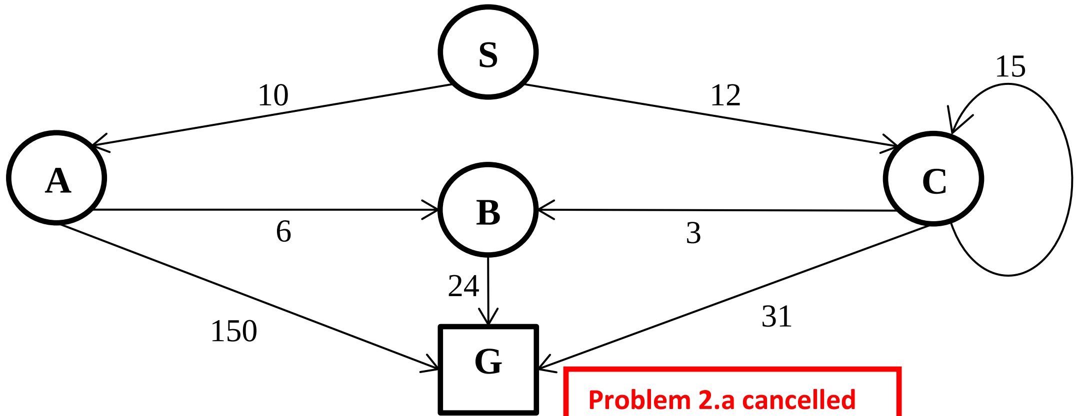
Task Environment	**Fully Observable or Partially Observable**	**Single Agent or Multi Agent**	**Deterministic or Stochastic**	**Episodic or Sequential**	**Static, Semi, or Dynamic**	**Discrete or Continuous**
Taxi driving robot	**Partially**	**Multi**	**Stochastic**	**Sequential**	**Dynamic**	**Continuous**
Crossword Puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess or Go with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker, bridge, blackjack etc.	Partially	Multi	Stochastic	Sequential	Static	Discrete
~~Part-picking~~ robot	~~Partially~~	~~Single~~	~~Stochastic~~	~~Episodic~~	~~Dynamic~~	~~Continuous~~

Cancelled because it was deemed confusing to students who had never before heard of a part-picking robot. Cancelled means that everyone gets it right, regardless of what you answered.

****** TURN PAGE OVER AND CONTINUE ON THE OTHER SIDE ******

2. (40 pts total, 8 pts each) STATE-SPACE SEARCH STRATEGIES. Execute Tree Search through this graph (i.e., do not remember visited nodes). Step costs are given next to each arc. The successors of each node are indicated by the directed arrows out of that node. **Successors are returned in left-to-right order, i.e., successors of S are (A, C), successors of A are (G, B), and successors of C are (B, G, C), in that order.** S is the only initial node, and G is the only goal node.

For each search strategy below, show the order in which nodes are expanded (i.e., to expand a node means that its children are generated), optionally ending with the goal node that is found, or indicate the repeating cycle if the search gets stuck in a loop. Show the path from start to goal, or write "None." Give the cost of the path that is found, or write "None." Do check for duplicate nodes in the Fringe/Frontier, and treat them appropriately. Do not check for loops. Do not check for duplicate nodes in Expanded.



Problem 2.a cancelled for reasons discussed in lecture. The solution shown here is correct.

~~**2.a. (8 pts total) DEPTH FIRST SEARCH.**~~

See Sections 3.4.3-4 and Figs. 3.16-17.

(6 pts) Order of node expansion: S A (G)

(1 pt) Path found: S A G **(1 pt)** Cost of path found: 160

2.b. (8 pts total) BREADTH FIRST SEARCH.

See Section 3.4.1 and Figs. 3.11-13.

(6 pts) Order of node expansion: S A (G)

(1 pt) Path found: S A G **(1 pt)** Cost of path found: 160

2.c. (8 pts total) UNIFORM COST S Fig. 3.24 Fig. 3.14

See Section 3.4.2 and Figs. 3.14-15.

(6 pts) Order of node expansion: S A C B B C B (G) or S A C B C B (G)

(1 pt) Path found: S C B G **(1 pt)** Cost of path found: 39

2.d. (8 pts total) ITERATED DEEPENING SEARCH.

See Sections 3.4.4-5 and Figs. 3.17-19.

(6 pts) Order of node expansion: S S A (G)

(1 pt) Path found: S A G **(1 pt)** Cost of path found: 160

2.e. (8 pts total) BIDIRECTIONAL SEARCH. Use Breadth First Search. First expand S, then expand G (invert the steps), then expand a node from Fringe(S), then expand a node from Fringe(G) (invert the steps), then expand a node from Fringe(S), then expand a node from Fringe(G) (invert the steps), and so on. **On the backward search from G, nodes are returned in right-to-left order (which is left-to-right if you stand on your head); i.e., successors of G are (C, B, A), successors of C are (C, S), successors of B are (C, A), and successors of A are (S, A), in that order on the backward search.**

(6 pts) Order of node expansion: S G (C)

See Sections 3.4.5 and Fig. 3.20.

(1 pt) Path found: S C G **(1 pt)** Cost of path found:

R&N Sec. 3.4.6 discusses the BDS termination condition for BFS. To clarify it, and to handle UCS:

*** For BFS, the search terminates when one fringe expands a node and discovers that one of the new children is present in the other fringe. This is quick and easy because the other fringe already maintains a hash table holding its fringe, as discussed in the lecture slides about removing duplicate nodes from the fringe, so you just look up the new child in the other fringe's hash table. If present, then you join the path from the Start to that child to the reverse of the path from the Goal to that child, and you have your path from Start to Goal.**

*** For UCS, the same applies, except that afterward you must continue searching until the sum of the costs of the nodes at the head of each queue is greater than or equal to the cost of the path you just found. This continuation guarantees that there is not a longer cheaper path somewhere in the queues. Of course, if you find a cheaper solution as the search winds down, it replaces the previous solution.**