CMSI 2130 – Classwork 2
SUPPLEMENT SOLUTION

Instructions:
This worksheet gives you some important practice with the basics of uninformed search:

- Provide answers to each of the following questions and write your responses in the blanks. If you are expected to show your work in arriving at a particular solution, space will be provided for you.
- Place the names of your group members below:

Group Members:

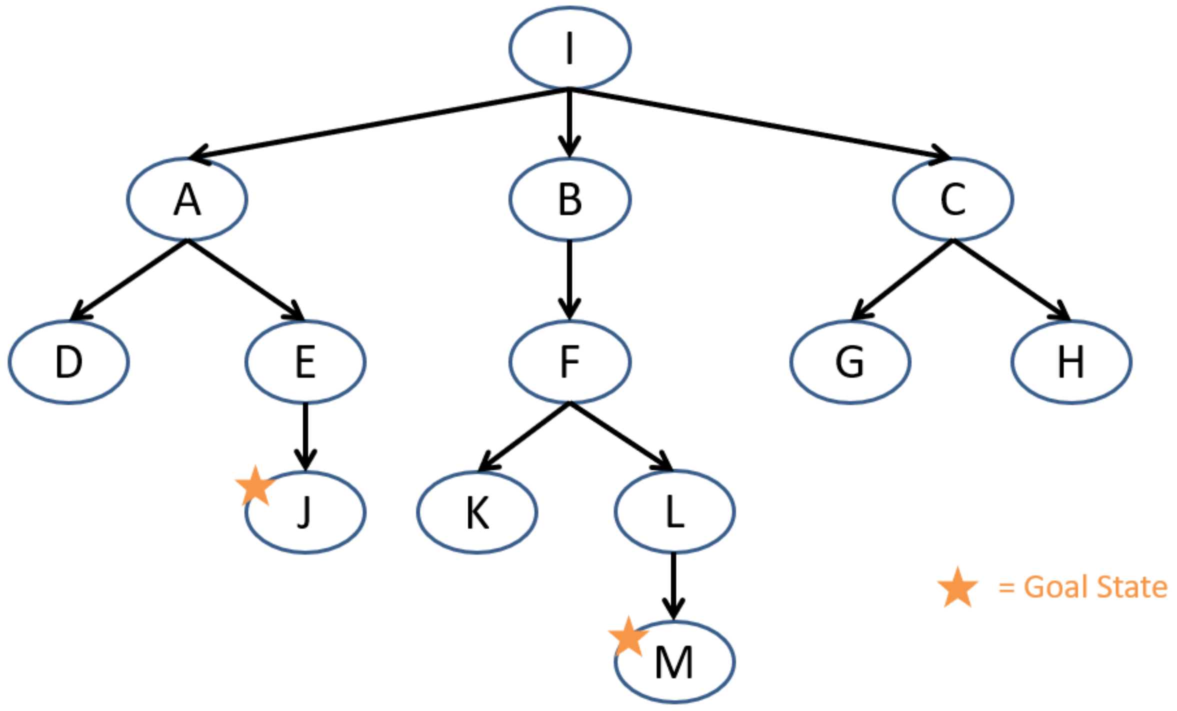
1. ______________________________
2. ______________________________
3. ______________________________

[!] NOTE: If completing digitally, since several exercises have you draw on an existing graphic, I suggest using editing tools on the PDF version (e.g., Adobe Acrobat or Mac's Preview). It is also acceptable to print this skeleton, write your solutions on the paper, and then scan your final copy *so long as* the resulting document is a single, multi-page PDF document that is legible (i.e., DO NOT submit like 10 blurry .jpgs you took with your 1998 Nokia).

Problem 1 – Uninformed Searches

Let's get some practice with the procedure of uninformed search strategies—the best way: doing a couple of examples by hand! Time to pretend you're a computer for a moment. Use the following search tree skeleton to answer the questions that follow—some notes:

- We'll provide some simple names to each node's state as capital letters A, B, C, etc. but this is assumed to be some arbitrary, abstract search problem.
- Goal states will be indicated with a star. Remember: with breadth-first and depth-first searches, the goal-test terminating condition is performed upon node *generation*.
- Although the full search tree is displayed for this worksheet, the following exercise traces *how that tree is constructed,* so recall that the full tree does not exist at the start (nor necessarily the end) of search, but is instead grown iteratively from the initial state / root. For example, node A doesn't exist until it is generated from having expanded its parent: node I.
- Node I, unimaginatively, represents the initial state from which all search strategies will begin.
- Assume that this is a uniform-cost problem in which each edge represents an action with the same cost as any other edge.



1.1. Using *Breadth-first Tree Search,* on the search tree from the previous page, complete the following questions.

1.1.1. Trace the execution of BFS by showing the state of the frontier and the next node expanded at each iteration of search in the table below. Some caveats:

- Remember that search terminates when the goal test finds a goal state, so you may not use the whole table below, but you also won't need more than its number of columns.
- In the event that multiple children are generated from an expanded parent, add them to the frontier in ascending alphabetic order. For example, when we expand the initial state, its children {A, B, C} are added to the frontier in order of A first (earliest alphabetically), *then* B, and finally C.
- The first two columns / iterations are done for you. Complete the remaining columns until the search would terminate.

Queue Back									
					H	J			
			E	F	G	H			
		C	D	E	F	G			
Queue Front		B	C	D	E	F			
	I	A	B	C	D	E			

1.1.2. In what order were nodes *expanded* during the search operation in 1.1.1.? (you may not use every cell below and should not include the found goal).

I	**A**	**B**	**C**	**D**	**E**			

1.1.3. Determine (1) which goal state was found by the search operation in 1.1.1. and (2) whether or not that goal is optimal and why.

(1) The goal at node J is found as soon as node E is expanded.
(2) This is the optimal goal because no goal node that has a lesser cost (i.e., for uniform-cost problems = closer to the root) exists in the tree.

1.2. Now using *Depth-limited Tree Search* with a generation depth limit $m = 4$ and the same search tree from the previous problem, answer the following questions:

1.2.1. Repeat exercise 1.1.1. in the table below, but for DLFS with a depth limit $m = 4$. Again, the first two columns / iterations are done for you, but note the change in the frontier's data structure!

Stack Top									
			H						
		C	G	G			L	M	
Stack Bottom		B	B	B	B	F	K	K	
	I	A	A	A	A	A	A	A	

1.2.2. In what order were nodes expanded during the search operation in 1.2.1.? (you may not use every cell below).

I	**C**	**H**	**G**	**B**	**F**	**L**		

1.2.3. Determine (1) which goal state was found by the search operation in 1.2.1. and (2) whether or not that goal is optimal and why.

(1) The goal at node M is found as soon as node L is expanded.
(2) This is *not* the optimal goal because a lower-cost goal at node J exists in the tree. DLFS found the suboptimal goal state here because its depth-limit $m = 4$ was greater than the depth of the optimal goal, and we used the alphabetical node generation order. Had it been set to $m = 3$, we would have found goal J instead. This goes to show the brittleness of depth-limited searches outside of the iteratively-deepening depth first search that would have avoided this issue.