Below, for each problem on this Midterm Exam, “Perfect” is the percentage of students who received full credit, “Partial” is the percentage who received partial credit, and “Zero” is the percentage who received zero credit.

(Due to rounding, values below may be only approximate estimates.)

We will supply these numbers as they become available.

CS-171, Intro to A.I. — Mid-term Exam — Winter Quarter, 2016

YOUR NAME: ____

YOUR ID: __________ ID TO RIGHT: __________ ROW: ________ SEAT: ________

The exam will begin on the next page. <u>Please, do not turn the page until told.</u>

When you are told to begin the exam, please check first to make sure that you have all ten pages, as numbered 1-10 in the bottom-right corner of each page. We wish to avoid copy problems. We will supply a new exam for any copy problems.

The exam is closed-notes, closed-book. No calculators, cell phones, electronics.

<u>Please turn off all cell phones now.</u>

Please clear your desk entirely, except for pen, pencil, eraser, a blank piece of paper (for scratch pad use), and an optional water bottle. Please write your name and ID# on the blank piece of paper and <u>turn it in with your exam</u>.

<u>This page summarizes the points for each question, so you can plan your time.</u>

1. (5 pts total, -1 pts for each error, but not negative) MINI-MAX SEARCH IN GAME TREES.

2. (10 pts total, -1 for each error, but not negative) ALPHA-BETA PRUNING.

3. (20 pts total) CONSTRAINT SATISFACTION PROBLEMS.

4. (15 pts total, -1 for each error, but not negative) CONSTRAINT SATISFACTION PROBLEMS.

5. (20 pts total, 5 pts each) STATE-SPACE SEARCH STRATEGIES.

6. (8 pts total, 1 pt each) LOCAL SEARCH --- SIMULATED ANNEALING.

7. (8 pts total, -1 pt each wrong answer, but not negative) SEARCH PROPERTIES.

8. (10 pts total, 1 pt each) ADVERSARIAL (GAME) SEARCH CONCEPTS.

9. (4 pts total, 1 pt each) TASK ENVIRONMENT.

<u>The Exam is printed on both sides to save trees! Work both sides of each page!</u>

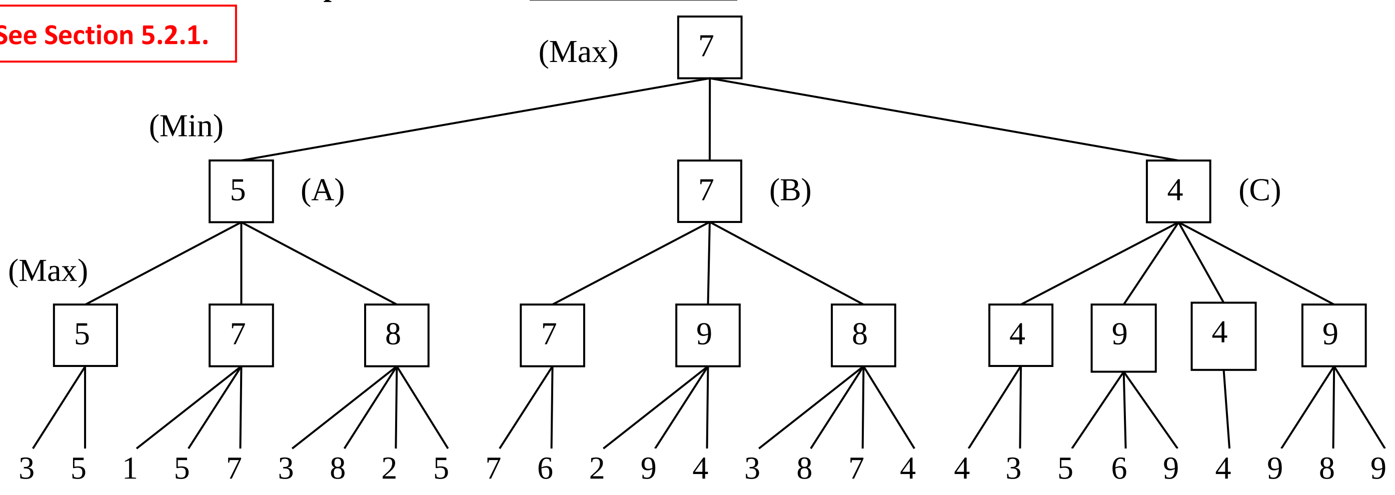
1. (5 pts total, -1 pts for each error, but not negative) MINI-MAX SEARCH IN GAME TREES. The game tree below illustrates a position reached in the game. Process the tree left-to-right. It is **Max**'s turn to move. At each leaf node is the estimated score returned by the heuristic static evaluator.

1.a. Fill in each blank square with the proper mini-max search value.

1.b. What is the best move for Max? (write A, B, or C) __B______

1.c. What score does Max expect to achieve? ________7______

See Section 5.2.1.

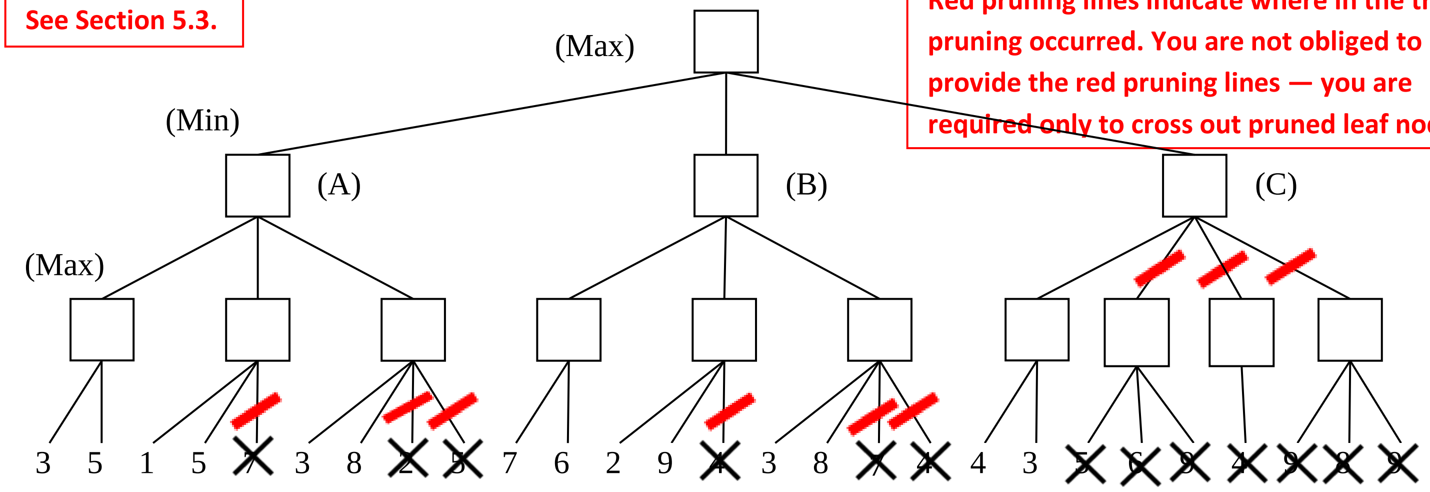


2. (10 pts total, -1 for each error, but not negative) ALPHA-BETA PRUNING. Process the tree left-to-right. This is the same tree as above (1.a). You do not need to indicate the branch node values again.

<u>Cross out each leaf node that will be pruned by Alpha-Beta Pruning. Do not just draw pruning lines.</u>

See Section 5.3.

Red pruning lines indicate where in the tree pruning occurred. You are not obliged to provide the red pruning lines — you are required only to cross out pruned leaf nodes.



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

3. (20 pts total) CONSTRAINT SATISFACTION PROBLEMS. You are a scheduling robot assigned to schedule professors and computer science classes that meet M/W/F. There are 5 classes to schedule and 3 professors to teach these classes. Your requirements are: (1) each professor only teaches one class at a time; (2) each class is taught by only one professor; and (3) some professors can only teach some of the classes. You must produce a complete and consistent schedule.

You decide to formulate this task as a CSP in which classes are the variables (named C1 through C5) and professors are the domain values (named A, B, and C). After you have solved the CSP, each class (variable) will be assigned one professor (value), and all constraints will be satisfied.

The classes (variables) are:

- C1, Class 1 - Intro to Programming: meets from 8:00-8:50am
- C2, Class 2 - Intro to Artificial Intelligence: meets from 8:30-9:20am
- C3, Class 3 - Natural Language Processing: meets from 9:00-9:50am
- C4, Class 4 - Computer Vision: meets from 9:00-9:50am
- C5, Class 5 - Machine Learning: meets from 9:30-10:20am

The professors (domain values) are:

- A, Professor A, who is available to teach Classes C3 and C4.
- B, Professor B, who is available to teach Classes C2, C3, C4, and C5.
- C, Professor C, who is available to teach Classes C1, C2, C3, C4, C5.

3.a (5 pts, 1 pt each) Variables and Domains. For each variable C1-C5 below, write down its domain as a subset of the values {A, B, C}. Enforce unary constraints as a preprocessing step, i.e., delete from the domain of each class variable any professor who is not available to teach that class.

3.a.1 Variable = **C1**, Domain = **{ C }**

3.a.2 Variable = **C2**, Domain = **{ B, C }**

3.a.3 Variable = **C3**, Domain = **{ A, B, C }**

3.a.4 Variable = **C4**, Domain = **{ A, B, C }**

3.a.5 Variable = **C5**, Domain = **{ B, C }**

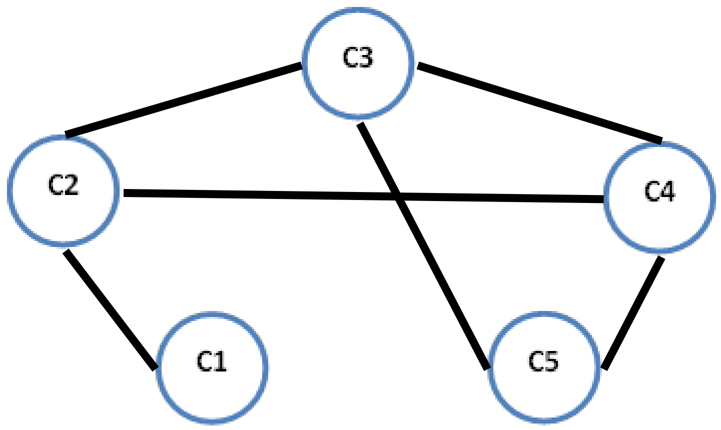
3.b (5 pts, -1 for each error, but not negative) Constraints. Write below all constraints that are associated with this CSP. Write each constraint **implicitly** as **Ci ≠ Cj** for all classes Ci and Cj that overlap in time and so cannot be taught by the same professors:

Write all CSP constraints as implicit constraints of the form Ci ≠ Cj:

C1 ≠ C2
C2 ≠ C3
C2 ≠ C4
C3 ≠ C4
C3 ≠ C5
C4 ≠ C5

****** PROBLEM CONTINUES ON THE NEXT PAGE ******

3.c (3 pts, -1 for each error, but not negative) Constraint Graph. Draw the constraint graph associated with your CSP. The nodes are provided for you. Draw the arcs.



3.d (5 pts, -1 for each error, but not negative) Arc Consistency (AC-3).
Run Arc Consistency (AC-3) on the domains in 3.a, the constraints in 3.b, and the constraint graph in 3.c. Write down the reduced domains that result when all inconsistent domain values are removed by Arc Consistency (AC-3).

Your answer will be considered correct if it correctly works Arc Consistency (AC-3) with respect to the answers you gave in 3.a, 3.b, and 3.c, even if those answers were wrong.

3.d.1 Variable = **C1**, AC-3 Reduced Domain = ______**{ C }**______

3.d.2 Variable = **C2**, AC-3 Reduced Domain = ______**{ B }**______

3.d.3 Variable = **C3**, AC-3 Reduced Domain = ______**{ A, C }**______

3.d.4 Variable = **C4**, AC-3 Reduced Domain = ______**{ A, C }**______

3.d.5 Variable = **C5**, AC-3 Reduced Domain = ______**{ B, C }**______

Note that C5 cannot possibly be C, but Arc Consistency does not rule it out. Path Consistency would rule it out, but costs more to run. Always trade-offs....

3.e (2 pts): Give one solution to this CSP. A solution is a complete and consistent assignment.

3.e.1 Variable = **C1**, Solution Value = ______**C**______

3.e.2 Variable = **C2**, Solution Value = ______**B**______

3.e.3 Variable = **C3**, Solution Value = ______**C**______

3.e.4 Variable = **C4**, Solution Value = ______**A**______

3.e.5 Variable = **C5**, Solution Value = ______**B**______

The other solution is:
C1 = C
C2 = B
C3 = A
C4 = C
C5 = B

Marvel at how little search actually needed to be done. C1 must be C and C2 must be B. C3 can be either A or C, and C4 must be the other. C5 must be B.

**** TURN PAGE OVER AND CONTINU

4. (15 pts total, -1 for each error, but not negative) CONSTRAINT SATISFACTION PROBLEMS. You are a robot assigned to place three Knights on a 3 x 3 chess board such that (1) all Knights do not attack each other, (2) only one Knight is placed in each row, and (3) only one Knight is placed in each column. Recall that a knight attacks in an "L" pattern, i.e., a knight moves two spaces rectilinearly and then one space at right angles to the initial movement.

For the benefit of students who do not play chess, the following boards illustrate the allowed cells (marked as **O**) and prohibited cells (marked as **X**) after you place a knight (marked as **K**) at a cell. Once a knight is placed at a cell marked as **K**, any other knight may be placed only in a cell marked as **O**, and may **not** be placed in a cell marked as **X**. Only symmetry-collapsed positions are shown below. All other constraint positions may be obtained by symmetry reflections and rotations.

row 3	**K**	**X**	**X**
row 2	**X**	**O**	**X**
row 1	**X**	**X**	**O**
	col C1	col C2	col C3

row 3	**X**	**O**	**X**
row 2	**K**	**X**	**X**
row 1	**X**	**O**	**X**
	col C1	col C2	col C3

row 3	**O**	**X**	**O**
row 2	**X**	**K**	**X**
row 1	**O**	**X**	**O**
	col C1	col C2	col C3

You decide to formulate this task as a CSP in which columns are the variables (named C1 through C3) and rows are the domain values (named 1, 2, and 3). After you have solved the CSP, each column (variable) will be assigned a row (value), and all constraints will be satisfied.

The columns (variables) are C1-C3:

- C1, col 1 - will be assigned the row number of the knight in column 1
- C2, col 2 - will be assigned the row number of the knight in column 2
- C3, col 3 - will be assigned the row number of the knight in column 3

The rows (domain values) are the digits 1-3:

- 1, the knight in that column is in row 1
- 2, the knight in that column is in row 2
- 3, the knight in that column is in row 3

There are no unitary constraints, and so the initial variable domains include all possible values:

D1 = {1, 2, 3} **D2 = {1, 2, 3}** **D3 = {1, 2, 3}**

4.a (5 pts total, -1 for each error, but not negative) Constraints. Write below all constraints that are associated with this CSP. Write each constraint **explicitly** as the list of **allowed** value pairs for that pair of variables. Note well that in problem **3.b** above you were asked to give **implicit** constraints, while here you are asked to list **explicit** constraints. Recall that a constraint is a pair: (1) a tuple that lists the variables involved and (2) a relation that lists the set of allowed combinations of domain values from those variables. For each tuple below, list explicitly the allowed pairs of domain values.

Write all CSP constraints as explicit relations of all allowed pairs of variable values:

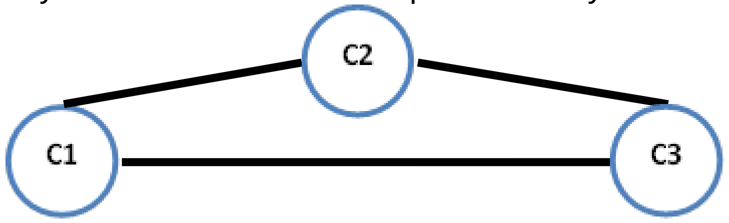
Tuple = **<C1, C2>**, Relation = { (1,2), (2,1), (2,3), (3,2) }

Tuple = **<C1, C3>**, Relation = { (1,3), (3,1) }

Tuple = **<C2, C3>**, Relation = { (1,2), (2,1), (2,3), (3,2) }

****** PROBLEM CONTINUES ON THE NEXT PAGE ******

4.b (2 pts total, -1 for each error, but not negative) Constraint Graph. Draw the constraint graph associated with your CSP. The nodes are provided for you. Draw the arcs.



4.c (3 pts, 1 pt each) Arc Consistency (AC-3). Run Arc Consistency (AC-3) as a preprocessing step on the domains shown above, the constraints in 4.a, and the constraint graph in 4.b. Write down the reduced domains that will result after running Arc Consistency (AC-3).

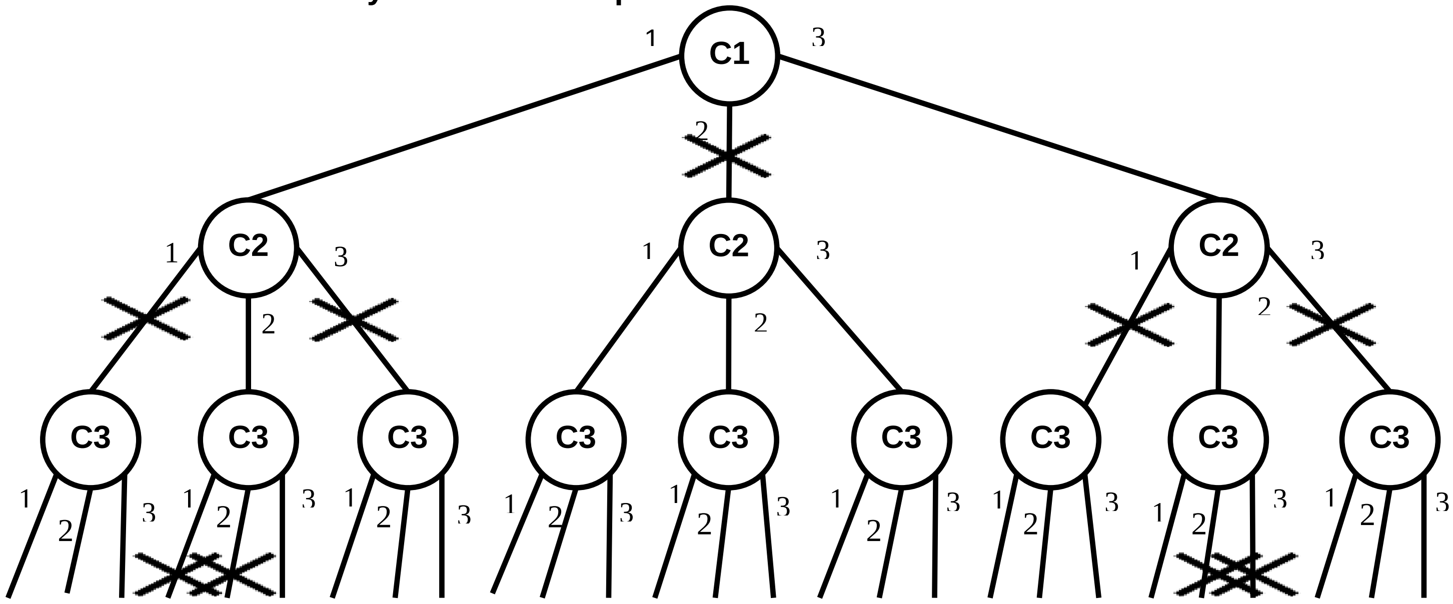
4.c.1 Variable = **C1**, AC-3 Reduced Domain = **{ 1, 3 }**

4.c.2 Variable = **C2**, AC-3 Reduced Domain = **{ 2 }**

4.c.3 Variable = **C3**, AC-3 Reduced Domain = **{ 1, 3 }**

Your answer will be considered to be correct if it is correct with respect to your answers in problems 3.a and 3.b, even if those answers were wrong.

4.d. (5 pts total, -1 for each error, but not negative) Backtracking Search, Forward Checking. Traverse the search tree below while doing Backtracking Search with Forward Checking, starting from the reduced domain shown in 4.c above. The full search tree, with all possibilities, is shown below. **Cross out any arc (edge) corresponding to a domain value that already was eliminated by constraint propagation (either by Forward Checking or by Arc Consistency).** At each leaf of your search tree, mark each failure as **X**, and mark each solution as **O**. Find all solutions. (Don't stop.)
Select variables in order (C1, C2, C3), and then select values in order (1, 2, 3).
The first one is done for you as an example.



X	X	X	X	X	O	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	O	X	X	X	X	X

At each leaf of your traversed search tree, mark each failure as **X**, and mark each solution as **O**.

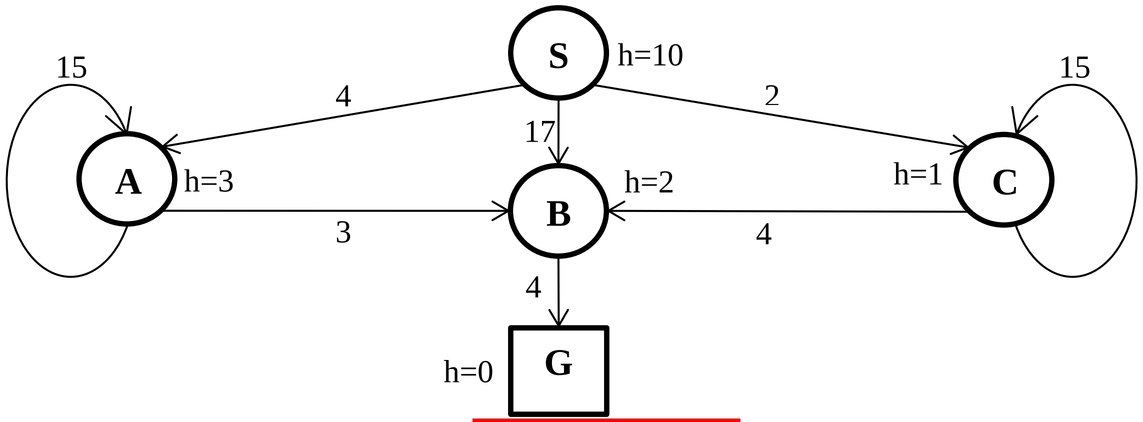
Marvel at how little search actually needed to be done. You marched directly to the two solutions, and never needed to backtrack.

****** TURN PAGE OVER AND CONTINUE ON THI**

5. (20 pts total, 5 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. Heuristic values are given next to each node (as h=x). The successors of each node are indicated by the arrows out of that node. Successors are returned in left-to-right order, i.e., successors of S are (A, B, C), successors of A are (A, B), and successors of C are (B, C), in that order.

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), 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"

The first one is done for you as an example.



5.a. DEPTH FIRST SEARCH.

See Section 3.4.3 and Fig. 3.17.

DFS does the Goal-test before the child is pushed onto the queue.

Order of node expansion: S A A A A A ...

Path found: None Cost of path found: None

5.b. (5 pts) UNIFORM COST SEA

UCS does goaltest when node is popped off queue.

See Section 3.4.2 and Fig. 3.14.

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

(2 pts) Path found: S C B G **(1 pt)** Cost of path found: 10

5.c. (5 pts) GREEDY (BEST-FIRST) SEARCH

GBFS has the same behavior whether the goaltest is done before node is pushed or after node is popped, because h=0 for a goal node, so goal nodes always sort to the front of the queue anyway.

(2 pts) Order of node expansion: S C C C C C ...

(2 pts) Path found: None

See Section 3.5.1 and Fig. 3.23.

(1 pt) Cost of path found: None

5.d. (5 pts) ITERATED DEEPENING SEARCH.

IDS does the Goal-test before the child is pushed onto the queue.

(2 pts) Order of node expansion: S S A B G

(2 pts) Path found: S B G

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

(1 pt) Cost of path found: 21

5.e. (5 pts) A* SEARCH.

A* does goaltest when node is popped off queue.

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

(2 pts) Path found: S C B G

See Section 3.5.2 and Figs. 3.24-25.

(1 pt) Cost of path found: 10

TECHNICAL NOTE: Technically, the goal node is not expanded, because no children of a goal node are generated. The goal node is listed in "Order of node expansion" for your convenience. Your answer is correct if you do not show the goal node in "Order of node expansion" — but it is a nicety to do so. Nevertheless, "Path found" *always* must show the goal node, because a path to a goal always must end in a goal.

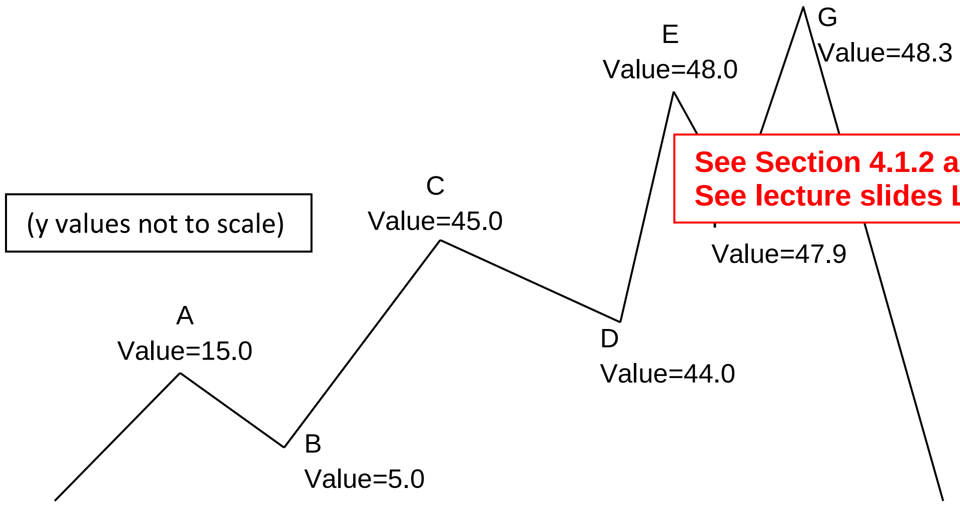
6. (8 pts total, 1 pt each) LOCAL SEARCH --- SIMULATED ANNEALING.
This question asks about Simulated Annealing local search. In the value landscape cartoon below, you will be asked about the probability that various moves will be accepted at different temperatures. Recall that Simulated Annealing always accepts a better move (∆Value = Value[next] — Value[current] > 0.0); but it accepts a worse move (∆Value < 0.0) only with probability e^(∆Value/T), where T is the current temperature on the temperature schedule.

Please use this temperature schedule (usually, it is a decaying exponential; but it is simplified here):

time (t)	1-100	101-200	201-300
Temperature (T)	10.0	1.0	0.1

You do not need a calculator; the values given have been chosen to follow this table:

x	0.0	-0.1	-0.4	-1.0	-4.0	-40.0
e^x	1.00	≈0.90	≈0.67	≈0.37	≈0.02	≈4.0e-18



See Section 4.1.2 and Figure 4.5.
See lecture slides Local Search.

Give your answer to two significant decimal places. **The first one is done for you as an example.**

6.a. (example) You are at Point A and t=23. **The probability you will accept a move A -> B =** 0.37

6.b. (1 pt) You are at Point B and t=23. **The probability you will accept a move B -> C =** 1.00

6.c. (1 pt) You are at Point C and t=123. **The probability you will accept a move C -> B =** 4.0e-18

6.d. (1 pt) You are at Point C and t=123. **The probability you will accept a move C -> D =** 0.37

6.e. (1 pt) You are at Point E and t=123. **The probability you will accept a move E -> D =** 0.02

6.f. (1 pt) You are at Point E and t=123. **The probability you will accept a move E -> F =** 0.90

6.g. (1 pt) You are at Point G and t=123. **The probability you will accept a move G -> F =** 0.67

6.h. (1 pt) You are at Point G and t=223. **The probability you will accept a move G -> F =** 0.02

6.i. (1 pt) With a very, very, very long slow annealing schedule, are you more likely, eventually in the long run, to wind up at point A or at point G? **(write A or G)** G .

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

7. (8 pts total, -1 pt each wrong answer, but not negative) SEARCH PROPERTIES.
Fill in the values of the four evaluation criteria for each search strategy shown. Assume a tree search where b is the finite branching factor; d is the depth to the shallowest goal node; m is the maximum depth of the search tree; C* is the cost of the optimal solution; step costs are identical and equal to some positive ε; and in Bidirectional search both directions use breadth-first search.

Note that these conditions satisfy all of the footnotes of Fig. 3.21 in you

See Figure 3.21.

Criterion	Complete?	Time complexity	Space complexity	Optimal?
Breadth-First	Yes	O(b^d)	O(b^d)	Yes
Uniform-Cost	Yes	O(b^(1+floor(C*/ε))) O(b^(d+1)) also OK	O(b^(1+floor(C*/ε))) O(b^(d+1)) also OK	Yes
Depth-First	No	O(b^m)	O(bm)	No
Iterative Deepening	Yes	O(b^d)	O(bd)	Yes
Bidirectional (if applicable)	Yes	O(b^(d/2))	O(b^(d/2))	Yes

8. (10 pts total, 1 pt each) ADVERSARIAL (GAME) SEARCH CONCEPTS. For each of the following terms on the left, write in the letter corresponding to the best answer or the correct definition on the right.

D	Game Strategy	A	Estiimates the value of a game state (i.e., of a game position)
H	Cut-off Test	B	In all game instances, total pay-off summed over all players is a constant
E	Alpha-Beta Pruning	C	Tree where nodes are game states and edges are game moves
G	Weighted Linear Function	D	Function that specifies a player's move in every possible game state
J	Terminal Test	E	Returns same move as MiniMax, but may prune more branches
I	Max	F	Optimal strategy for 2-player zero-sum games of perfect information, but impractical given limited time to make each move
C	Game Tree	G	Vector dot product of a weight vector and a state feature vector
A	Heuristic Evaluation Function	H	Function that decides when to stop exploring this search branch
B	Zero-sum Game	I	The player that tries to achieve higher scores
F	MiniMax Algorithm	J	Function that says when the game is over

9. (4 pts total, 1 pt 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 components.

P<u>erformance (measure)</u> E<u>nvironment</u> A<u>ctuators</u> S<u>ensors</u>

****** THIS IS THE END OF THE MID-TERM EXAM ******