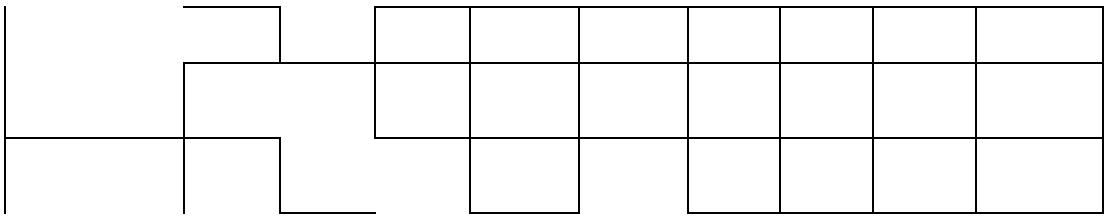
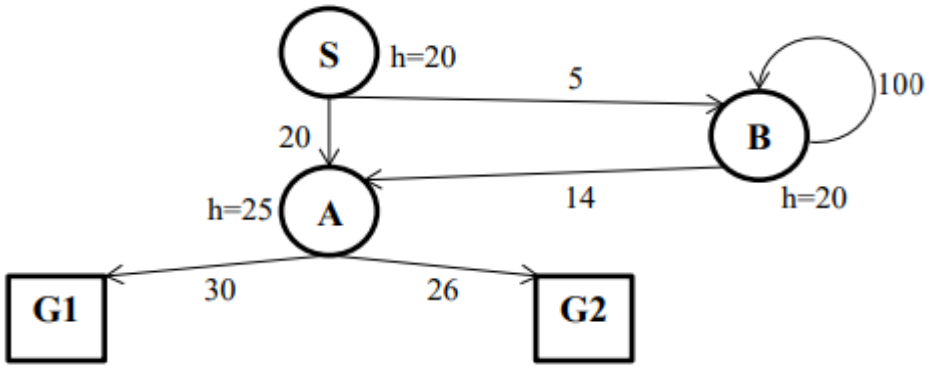


Question No. 1 SEARCH (12).

Execute Tree Search through this graph (i.e., do not remember visited nodes, so repeated nodes are possible). It is not a tree but pretend that you don't know that. 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. (Note: B is a successor of itself).

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. Show the path from start to goal or write "None". The first one is done for you as an example.



a) (02 pts) DEPTH FIRST SEARCH.

Order of node expansion: S A G1

Path found: S A G1

b) (02 pts) BREADTH FIRST SEARCH.

Order of node expansion: S A G1

Path found: S A G1

c) (02 pts) UNIFORM COST SEARCH.

Order of node expansion: S B A A G2

Path found: S B A G2

d) (02 pts) GREEDY (BEST-FIRST) SEARCH.

Order of node expansion: S B B B B B B ... etc.

Path found: None

e) (02 pts) ITERATED DEEPENING SEARCH.

Order of node expansion: S S A G1

Path found: S A G1

f) (02 pts) A* SEARCH.

Order of node expansion: S B A A G2

Path found: S B A G2

Question No. 2 STATE SPACE SEARCH (10).

The ***Missionaries and Cannibals*** problem is a classic brainteaser. It is a member of a class of river-crossing brainteasers that includes *Farmer to Market, Jealous Husbands,* and others *Three missionaries and three cannibals must cross a river using a boat which can carry at most two people. The constraint is that, for both riverbanks, if there are missionaries present on the bank, they cannot be outnumbered by cannibals (if they were, the cannibals would eat the missionaries). The boat cannot cross the river by itself without people on board.* **How can all six people get safely across the river?**

We will call states where anyone gets eaten "forbidden" and consider them as part of the state space, but from which there is no return; i.e., there is no way to go to any other state (including back to the previous state) from such a forbidden state. All other states are "allowed." It is always possible to go from an allowed state to some other state(s).

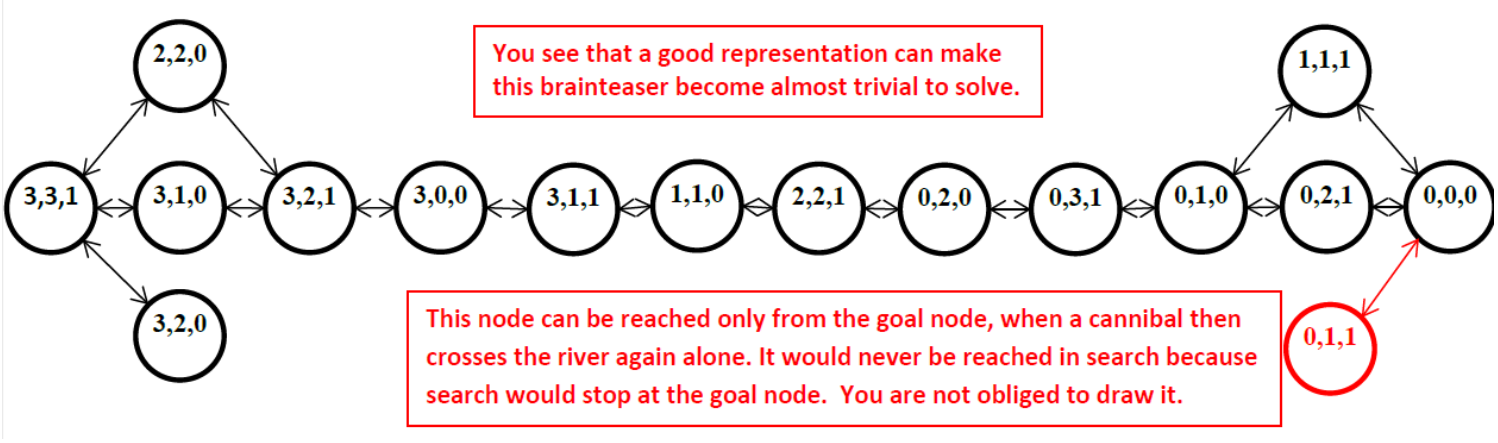
One way to represent this problem is as a state vector with components (M,C,B), where M=the number of missionaries on the wrong side, C=the number of cannibals on the wrong side, and B=the number of boats on the wrong side. Since everything starts on the wrong side, the start state is (3,3,1). The single goal state is (0,0,0).

a) **(2 pts)** What is the maximum branching factor counting BOTH allowed and forbidden states as children?

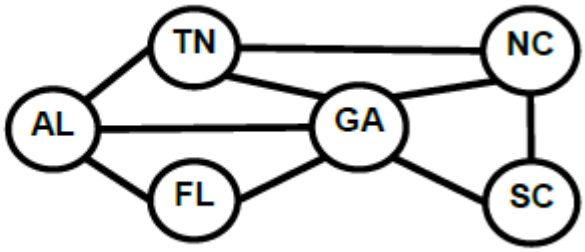
5

.

b) **(8 pts) Draw the state space showing ONLY the ALLOWED states. I.e., do NOT show the FORBIDDEN states.**
Represent an allowed state as a circle enclosing its state vector (M, C, B). Connect allowed states that are possible successors of each other. The start and goal states are shown. The first node expansion is done for you as an example.



Question No. 3 CONSTRAINT SATISFACTION PROBLEMS. (20).



You are a map-coloring robot assigned to color this Southeast USA map. Adjacent regions must be colored a different color (R=Red, B=Blue, G=Green). The constraint graph is shown.

a) (4pts) FORWARD CHECKING. Cross out all values that would be eliminated by Forward Checking, after variable GA has just been assigned value G, as shown:

AL	TN	FL	GA	NC	SC
R ~~G~~ B	R ~~G~~ B	R ~~G~~ B	G	R ~~G~~ B	R ~~G~~ B

b) (4pts) ARC CONSISTENCY. AL and FL have been assigned values, but no constraint propagation has been done. Cross out all values that would be eliminated by Arc Consistency

AL	TN	FL	GA	NC	SC
B	R ~~G~~ ~~B~~	R	~~R~~ G ~~B~~	~~R~~ ~~G~~ B	R ~~G~~ ~~B~~

c) (4pts) MINIMUM-REMAINING-VALUES HEURISTIC. Consider the assignment below. TN is assigned and constraint propagation has been done. List all unassigned variables that might be selected by the Minimum-Remaining-Values (MRV) Heuristic: **AL, GA, NC**

AL	TN	FL	GA	NC	SC
R B	G	R G B	R B	R B	R G B

d) (4pts) DEGREE HEURISTIC. Consider the assignment below. (It is the same assignment as in problem 6c above.) TN is assigned and constraint propagation has been done. List all unassigned variables that might be selected by the Degree Heuristic: **GA**

AL	TN	FL	GA	NC	SC
R B	G	R G B	R B	R B	R G B

e) (4pts) MIN-CONFLICTS HEURISTIC. Consider the complete but inconsistent assignment below. GA has just been selected to be assigned a new value during local search for a complete and consistent assignment. What new value would be chosen below for GA by the Min-Conflicts Heuristic? _____**R**_____

AL	TN	FL	GA	NC	SC
B	G	R	?	G	B

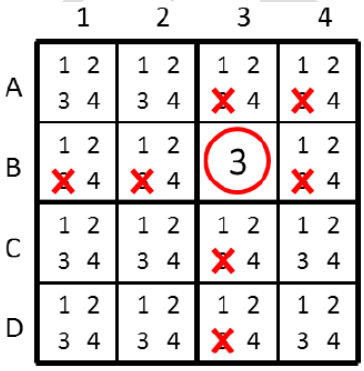
Question No. 4 SUDOKU AS A CONSTRAINT SATISFACTION PROBLEM (12).

A Sudoku board consists of $n \times n$ squares, some of which are initially filled with digits from 1 to n. The objective of the Sudoku puzzle is to fill in all the remaining squares such that no digit appears twice in any row, column, or $\sqrt{n} \times \sqrt{n}$ box. **Consider the case in which $n = 4$.**

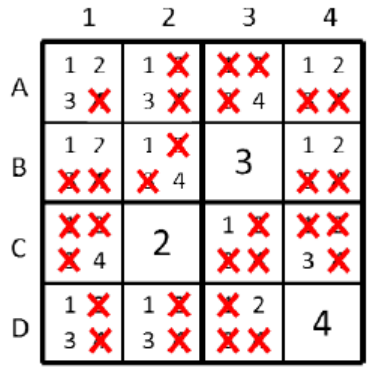
The Sudoku puzzle with $n = 4$ can be formulated as a constraint satisfaction problem with $n^2 = 16$ variables, where every variable stands for one of the 16 squares on the 4x4 Sudoku board. If we denote rows with letters A-D and columns with numbers 1-4, we end up with 16 variables: A1, A2, A3, A4, B1, B2, …, D3, D4. Each variable has domain values {1, 2, 3, 4}.

What remains to be done is to specify the constraints of the problem. We could use binary constraints for each pair of variables but might end up with a large number of constraints. A better alternative is to use global *A*llDiff constraints, where *A*llDiff (*X*1, *X*2, …, *Xn*) means that all n variables, *X*1, *X*2, …, *Xn*, must have different values.

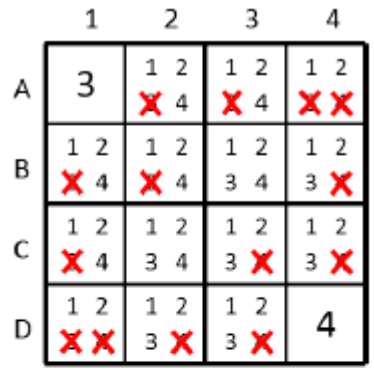
a) (2 pts) How many ***A*llDiff** constraints are required to specify that no digit may appear twice in any row, column, or **2** × **2** box? _**12 (= 1 per row, 1 per column, and 1 per box, = 3 constraints across 4 cases)** ___



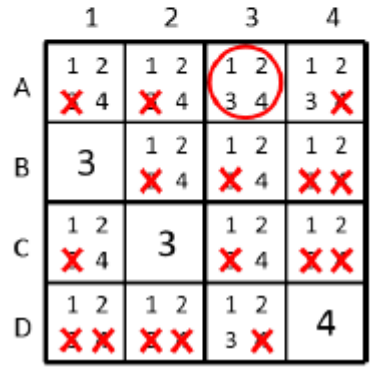
b) (2 pts) FORWARD CHECKING.
Consider the 4×4 Sudoku board on the left. Possible domain values for each variable are shown in each square. Variable B3 has just been assigned value 3 (circled).
Cross out all values from the domains of the remaining unassigned variables that now would be eliminated by Forward Checking.



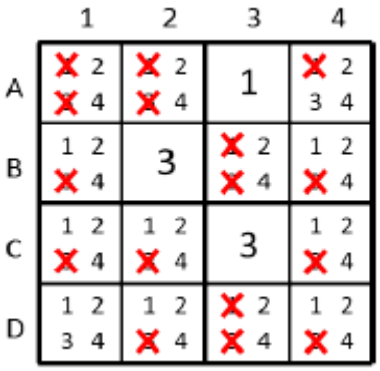
c) (2 pts) ARC CONSISTENCY.
Consider the 4×4 Sudoku board on the left. Possible domain values for each variable are shown in each square. Variables B3, C2, and D4 have been assigned values, but no constraint propagation has been done.
Cross out all values from the domains of the remaining unassigned variables that now would be eliminated by Arc Consistency



d) (2 pts) MINIMUM-REMAINING-VALUES (MRV) HEURISTIC.
Consider the 4×4 Sudoku board on the left. Possible domain values for each variable are shown in each square. Variables A1 and D4 are assigned and constraint propagation has been done.
List all unassigned variables (in any order) that now might be selected by the Minimum-Remaining-Values (MRV) Heuristic: A4,D1



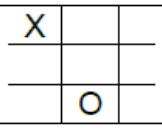
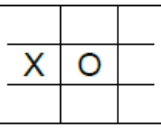
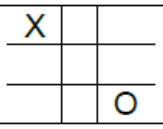
e) (2 pts) LEAST-CONSTRAINING-VALUE (LCV) HEURISTIC.
Consider the 4×4 Sudoku board on the left. Possible domain values for each variable are shown in each square. Variables B1, C2, and D4 are assigned and constraint propagation has been done. A3 (circled) is chosen for assignment next.
List all values (in any order) for variable A3 (circled) that now might be selected by the Least-Constraining-Value (LCV) Heuristic: 3



f. (2 pts) DEGREE HEURISTIC (DH).
Consider the 4×4 Sudoku board on the left. Possible domain values for each variable are shown in each square. Variables A3, B2, and C3 are assigned and constraint propagation has been done.
List all unassigned variables (in any order) that now might be selected by the Degree Heuristic (ignore MRV for this problem): D1

Question No. 5 GAME SEARCH WITH TIC-TAC-TOE AND WIN-PATHS HEURISTIC FUNCTION (18).

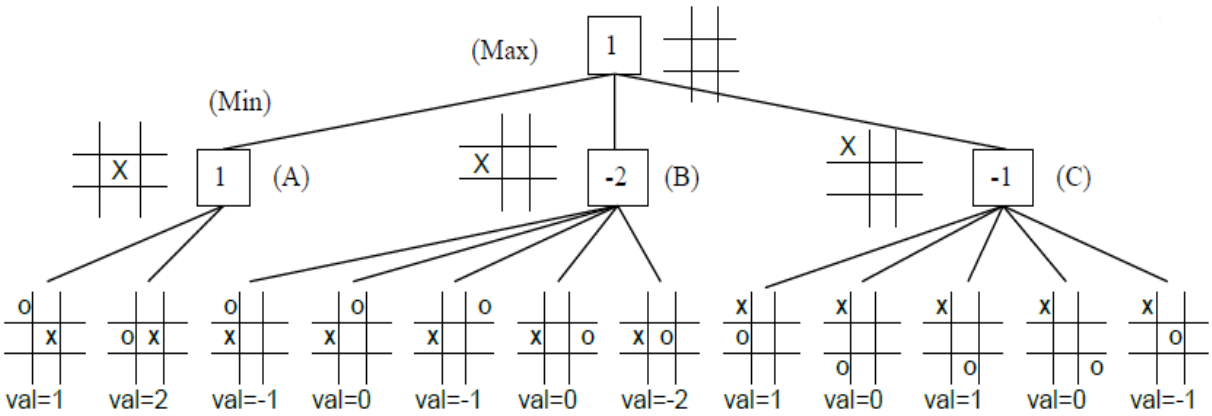
This problem asks about MiniMax Search and Alpha-Beta pruning in Tic-Tac-Toe with the Win-paths static heuristic evaluation function. Recall that the Win-paths heuristic function counts the number of possible win-paths for MAX (= X) and subtracts the number of possible win-paths for MIN (= O). For example:

		
MAX (= X) # win paths=6 MIN (= O) # win paths=5 Heuristic value = 1 (= 6-5)	MAX (= X) # win paths=4 MIN (= O) # win paths=6 Heuristic value = -2 = (4-6)	MAX (= X) # win paths=5 MIN (= O) # win paths=5 Heuristic value = 0 (= 5-5)

a) (6 pts total, 1 pt each blank branch node [4 in part 1] or answer space [1 each in parts 2&3])

In the game tree below it is **Max**'s (= X's) turn to move. At each leaf node is the estimated score of that resulting position as returned by the Win-path heuristic static evaluator (written below as "val=*n*").

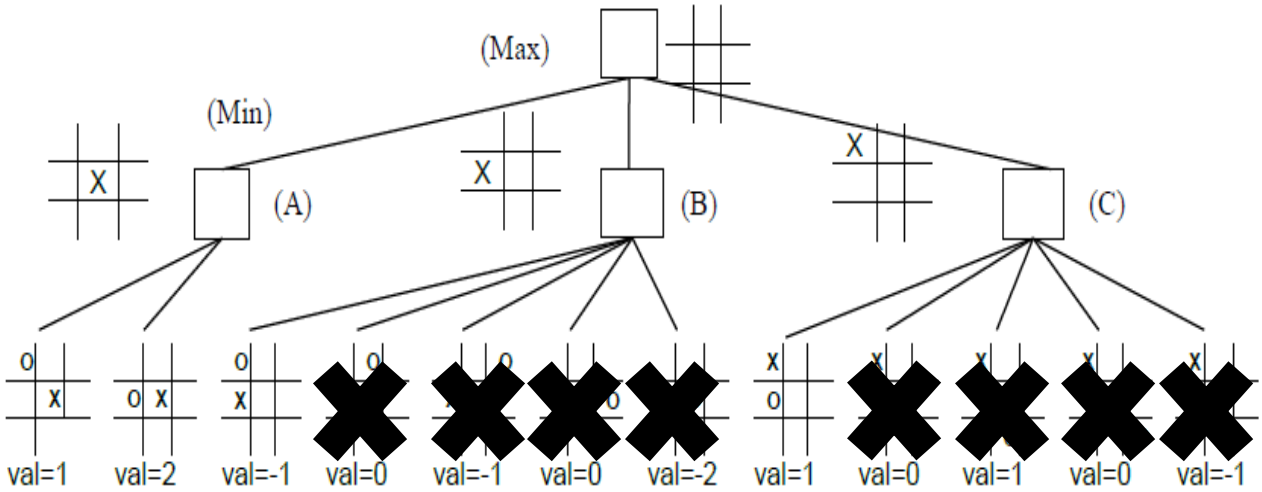
(1) Perform Mini-Max search and label each branch node with its return value (4 branch nodes).



(2) What is Max's best move (A, B, or C)? A

(3) What value does Max expect to get? 1

(12 pts total, 1 pt each leaf node)
In the game tree below it is **Max**'s (= X's) turn to move (this is the same game tree as in problem 6.a above). At each leaf node is the estimated score of that resulting position as returned by the Win-path heuristic static evaluator (as "val=*n*"). You do not need to indicate the branch node return values again.
Cross out each leaf value that would be pruned by Alpha-Beta Pruning.



Question No. 6 RESOLUTION THEOREM PROVING (10).

You are engaged in Knowledge Engineering for the Wumpus Cave. You have interviewed an expert on the Wumpus Cave who told you, among other things, "A breeze in square (1,1) is equivalent to a pit in square (1,2) or a pit in square (2,1)." You translated this into propositional logic as, "(B11 ⇔ P12 ∨ P21)," and then into Conjunctive Normal Form as "(¬B11 ∨ P12 ∨ P21) ∧ (¬P12 ∨ B11) ∧ (¬P21 ∨ B11)."
Now it is time for the first "live" test of your system. An agent has been lowered down into the Wumpus cave, and reports back by radio, "Square (1,1) has a breeze. Also, I went into square (1,2) and I did not die, so it does not have a pit." You translate this knowledge into propositional logic as "(B11) ∧ (¬ P12)" and add it to your knowledge base.
Next your system is asked to perform inference. The agent asks by radio, "Is it true that square (2,1) has a pit?" You translate this query into propositional logic as the goal sentence "(P21)." You form the negated goal as "(¬ P21)." Your knowledge base plus negated goal is:

(¬B11 ∨ P12 ∨ P21)
(¬P12 ∨ B11)
(¬P21 ∨ B11)
(B11)
(¬ P12)
(¬ P21)

Run resolution on this knowledge base until you produce the null clause, "( )", thereby proving that the goal sentence is true. The shortest proof I know of is only three lines long. It is OK to use more lines, if your proof is correct. SHOW YOUR WORK.
Repeatedly choose two clauses, write one clause in the first blank space on a line, and the other clause in the second. Apply resolution to them. Write the resulting clause in the third blank space and insert it into the knowledge base.

Resolve (¬B11 ∨ P12 ∨ P21) **and** (B11) **to give** (P12 ∨ P21)

Resolve (P12 ∨ P21) **and** (¬P12) **to give** (P21)

Resolve (P21) **and** (¬P21) **to give** ()

Other proofs are OK as long as they are correct. For example, you might perform the resolution steps above in any other order you choose.

Question No. 7 ONE FISH, TWO FISH, RED FISH, BLUE FISH (08).

Amy, Betty, Cindy, and Diane went out to lunch at a seafood restaurant. Each ordered one fish. Each fish was either a red fish or a blue fish. **Among them they had exactly two red fish and two blue fish.**

You translate this fact into Propositional Logic (in prefix form) as:
/* Ontology: Symbol A/B/C/D means that Amy/Betty/Cindy/Diane had a red fish. */
(or (and A B (¬ C) (¬ D)) (and A (¬ B) C (¬ D))
(and A (¬ B) (¬ C) D) (and (¬ A) B C (¬ D))
(and (¬ A) B (¬ C) D) (and (¬ A) (¬ B) C D)))

Their waiter reported:
"Amy, Betty, and Cindy had exactly one red fish among them; I don't remember who had what. Betty, Cindy, and Diane had exactly one red fish among them; I don't remember who had what."
You translate these facts into Propositional Logic (in prefix form) as:
(or (and A (¬ B) (¬ C)) (and (¬ A) B (¬ C)) (and (¬ A) (¬ B) C))
(or (and B (¬ C) (¬ D)) (and (¬ B) C (¬ D)) (and (¬ B) (¬ C) D))
Betty's daughter asked, "Is it true that my mother had a blue fish?"
You translate this query into Propositional Logic as "(¬ B)" and form the negated goal as "(B)".
Your resulting knowledge base (KB) plus the negated goal (in CNF clausal form) is:

(A B C) ((¬ A) (¬ B) (¬ C))
(A B D) ((¬ A) (¬ B) (¬ D))
(A C D) ((¬ A) (¬ C) (¬ D))
(B C D) ((¬ B) (¬ C) (¬ D))
((¬ A) (¬ B)) ((¬ A) (¬ C))
((¬ B) (¬ C)) ((¬ B) (¬ D))
((¬ C) (¬ D)) (B)

Write a resolution proof that Betty had a blue fish.
For each step of the proof, fill in the first two blanks with CNF sentences from KB that will resolve to produce the CNF result that you write in the third (resolvent) blank. The resolvent is the result of resolving the first two sentences. Add the resolvent to KB, and repeat. Use as many steps as necessary, ending with the empty clause. The empty clause indicates a contradiction, and therefore that KB entails the original goal sentence.
The shortest proof I know of is only four lines long. (A Bonus Point is offered for a shorter proof.) Longer proofs are OK provided they are correct. *Think about it, then find a proof that mirrors how you think.*
Resolve **(A C D)** with **((¬ A) (¬ B))** to produce: **((¬ B) C D)**
Resolve **((¬ B) C D)** with **((¬ B) (¬ C))** to produce: **((¬ B) D)**
Resolve **((¬ B) D)** with **((¬ B) (¬ D))** to produce: **(¬ B)**
Resolve **(¬ B)** with **(B)** to produce: **( )**

Other proofs are OK provided that they are correct.

Question No. 8 ENGLISH TO FOL CONVERSION (10).

For each English sentence below, write the FOL sentence that best expresses its intended meaning.

The preferred encoding is given first, followed by acceptable syntactic variants.

a) (1pt) "All persons are mortal." [Use: Person(x), Mortal (x)]

∀x Person(x) ⇒ Mortal(x)
∀x ¬Person(x) ∨ Mortal(x)
Common Mistakes:
∀x Person(x) ∧ Mortal(x)

b) (1pt) "Fifi has a sister who is a cat." [Use: Sister(Fifi, x), Cat(x)]

∃x Sister(Fifi, x) ∧ Cat(x)
Common Mistakes:
∃x Sister(Fifi, x) ⇒ Cat(x)

c) (1pt) "For every food, there is a person who eats that food." [Use: Food(x), Person(y), Eats(y, x)]

∀x ∃y Food(x) ⇒ [Person(y) ∧ Eats(y, x)]
∀x Food(x) ⇒ ∃y [Person(y) ∧ Eats(y, x)]
∀x ∃y ¬Food(x) ∨ [Person(y) ∧ Eats(y, x)]
∀x ∃y [¬Food(x) ∨ Person(y)] ∧ [¬ Food(x) ∨ Eats(y, x)]
∀x ∃y [Food(x) ⇒ Person(y)] ∧ [Food(x) ⇒ Eats(y, x)]
Common Mistakes:
∀x ∃y [Food(x) ∧ Person(y)] ⇒ Eats(y, x)
∀x ∃y Food(x) ∧ Person(y) ∧ Eats(y, x)

d) (1pt) "Every person eats every food." [Use: Person (x), Food (y), Eats(x, y)]

∀x ∀y [Person(x) ∧ Food(y)] ⇒ Eats(x, y)
∀x ∀y ¬Person(x) ∨ ¬Food(y) ∨ Eats(x, y)
∀x ∀y Person(x) ⇒ [Food(y) ⇒ Eats(x, y)]
∀x ∀y Person(x) ⇒ [¬Food(y) ∨ Eats(x, y)]
∀x ∀y ¬Person(x) ∨ [Food(y) ⇒ Eats(x, y)]
Common Mistakes:
∀x ∀y Person(x) ⇒ [Food(y) ∧ Eats(x, y)]
∀x ∀y Person(x) ∧ Food(y) ∧ Eats(x, y)

e) (2 pts) "All greedy kings are evil." [Use: King(x), Greedy(x), Evil(x)]

∀x [Greedy(x) ∧ King(x)] ⇒ Evil(x)
∀x ¬Greedy(x) ∨ ¬King(x) ∨ Evil(x)
∀x Greedy(x) ⇒ [King(x) ⇒ Evil(x)]
Common Mistakes:
∀x Greedy(x) ∧ King(x) ∧ Evil(x)

f) (1pt) "Everyone has a favorite food." [Use: Person(x), Food(y), Favorite(y, x)]

∀x ∃y Person(x) ⇒ [Food(y) ∧ Favorite(y, x)]
∀x Person(x) ⇒ ∃y [Food(y) ∧ Favorite(y, x)]
∀x ∃y ¬Person(x) ∨ [Food(y) ∧ Favorite(y, x)]
∀x ∃y [¬Person(x) ∨ Food(y)] ∧ [¬Person(x) ∨ Favorite(y, x)]
∀x ∃y [Person(x) ⇒ Food(y)] ∧ [Person(x) ⇒ Favorite(y, x)]
Common Mistakes:
∀x ∃y [Person(x) ∧ Food(y)] ⇒ Favorite(y, x)
∀x ∃y Person(x) ∧ Food(y) ∧ Favorite(y, x)

g) (1pt) "There is someone at UCI who is smart." [Use: Person(x), At(x, UCI), Smart(x)]

∃x Person(x) ∧ At(x, UCI) ∧ Smart(x)
Common Mistakes:
∃x [Person(x) ∧ At(x, UCI)] ⇒ Smart(x)

h) (1pt) "Everyone at UCI is smart." [Use: Person(x), At(x, UCI), Smart(x)]

∀x [Person(x) ∧ At(x, UCI)] ⇒ Smart(x)
∀x ¬[Person(x) ∧ At(x, UCI)] ∨ Smart(x)
∀x ¬Person(x) ∨ ¬At(x, UCI) ∨ Smart(x)
Common Mistakes:
∀x Person(x) ∧ At(x, UCI) ∧ Smart(x)
∀x Person(x) ⇒ [At(x, UCI) ∧ Smart(x)]

i) (1pt) "Every person eats some food." [Use: Person (x), Food (y), Eats(x, y)]

∀x ∃y Person(x) ⇒ [Food(y) ∧ Eats(x, y)]
∀x Person(x) ⇒ ∃y [Food(y) ∧ Eats(x, y)]
∀x ∃y ¬Person(x) ∨ [Food(y) ∧ Eats(x, y)]
∀x ∃y [¬Person(x) ∨ Food(y)] ∧ [¬Person(x) ∨ Eats(x, y)]
Common Mistakes:
∀x ∃y [Person(x) ∧ Food(y)] ⇒ Eats(x, y)
∀x ∃y Person(x) ∧ Food(y) ∧ Eats(x, y)

j) (1pt) "Some person eats some food." [Use: Person (x), Food (y), Eats(x, y)]

∃x ∃y Person(x) ∧ Food(y) ∧ Eats(x, y)

Common Mistakes:

∃x ∃y [Person(x) ∧ Food(y)] ⇒ Eats(x, y)