1. For each given set of sentences, determine whether it is logically consistent (jointly satisfiable).
   (a) $\neg D, (D \vee F), \neg F$
   (b) $(T \rightarrow K), \neg K, (K \vee \neg T)$
   (c) $\neg(A \rightarrow (\neg C \rightarrow B)), ((B \vee C) \wedge A)$
   (d) $(C \rightarrow B), (D \vee C), \neg B, (D \rightarrow B)$

2. Complete the following deductive formal proofs by filling in missing formulae and justifications.

   (a)

1	$H \rightarrow (R \wedge C)$	Premise	
2	$\neg R \vee \neg C$	Premise	
3	$\neg(R \wedge C)$		
$\therefore$		MT 1, 3	

   (b)

1	$K \wedge S$	Premise	
2	$\neg K$	Premise	
3			
4			
$\therefore$	$\neg S$		

   (c)

1	$A \rightarrow \neg A$	Premise	
$\vdots$		(multiple lines)	
$\therefore$	$\neg A$	LEM	

   (d)

1	$(P \wedge Q) \vee (P \wedge R)$	Premise	
2		Assumption	
3	P		
4		Assumption	
5	P		
$\therefore$	P		

3. Symbolize the given arguments with well-formed formulae (WFFs) of propositional logic. For each argument, determine its validity using a truth table. For each *valid* argument, provide a deductive formal proof[1] in Fitch notation. For each *invalid* argument, provide a counterexample valuation.
   (a) *If philosophers ponder profound problems, their quandaries quell quotidian quibbles. Either their quandaries don't quell quotidian quibbles or right reasoning reveals reality (or both). Philosophers do ponder profound problems. Therefore, right reasoning reveals reality.*
   (b) *If aardvarks are adorable, then either baby baboons don't beat bongos or crocodiles can't consume cute capybaras (or both). Baby baboons beat bongos. Aardvarks aren't adorable unless crocodiles can't consume cute capybaras. Therefore, aardvarks aren't adorable.*
   (c) *If discipline doesn't defeat deficiency, then geniuses generally get good grades. If discipline defeats deficiency, then homework has harmed humanity. Therefore, geniuses generally get good grades unless homework has harmed humanity.*
   (d) *Crocodiles can consume cute capybaras only if incarcerating iguanas isn't illegal. Mad monkeys make mayhem and dinosaurs do disco dance, unless crocodiles consume cute capybaras. It is known that incarcerating iguanas is illegal. Therefore, dinosaurs do disco dance if and only if mad monkeys make mayhem.*

4. For each given argument, construct a deductive proof in Fitch notation using only basic rules.
   (a) $\neg\neg A \therefore A$
   (b) $((A \rightarrow B) \rightarrow A) \therefore A$
   (c) $(\neg B \rightarrow \neg A) \therefore (A \rightarrow B)$
   (d) $\neg(A \vee B) \therefore (\neg A \wedge \neg B)$
   (e) $(\neg A \wedge \neg B) \therefore \neg(A \vee B)$
   (f) $(A \rightarrow B) \wedge (\neg A \rightarrow B) \therefore B$

5. For each given tautology, construct a deductive proof in Fitch notation.
   (a) $(A \rightarrow B) \vee (B \rightarrow A)$
   (b) $A \rightarrow (B \rightarrow A)$
   (c) $(\neg B \rightarrow \neg A) \rightarrow ((\neg B \rightarrow A) \rightarrow B)$
   (d) $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$

[1] You can check your proofs at https://proofs.openlogicproject.org. Note that some inference rules may be missing here, *e.g.*, contraposition and commutativity—nevertheless, you are still allowed to use them in this task.

6. Reduce *any three*[2] of the following problems to the Boolean satisfiability problem (SAT). Provide a detailed encoding of each chosen problem into logical variables and propositional constraints. While your encoding does not have to be in CNF, explain how high-level constraints (such as arithmetic conditions) translate into propositional logic. Additionally, discuss possible extensions or variations for each problem, and describe how your reduction could be adapted to handle these cases effectively.
    0. *(Do not pick this one!)* **Graph Coloring:** Determine if a given graph $G = (V, E)$ can be properly colored with k colors so that no two adjacent vertices share the same color.
    1. **Sudoku Puzzle:** Determine if a partially filled 9×9 Sudoku grid can be completed so that each row, column, and 3×3 sub-grid contains each digit from 1 to 9 exactly once.
    2. **N-Queens Problem:** Place N queens on an $N \times N$ chessboard so that no two queens threaten each other (no shared row, column, or diagonal).
    3. **Hamiltonian Cycle:** Determine if a given directed graph $G = (V, E)$ contains a Hamiltonian cycle that visits each vertex exactly once and returns to the starting point.
    4. **Clique:** Determine if a graph $G = (V, E)$ has a k-clique: a complete subgraph on k vertices.
    5. **Vertex Cover:** Determine if a graph $G = (V, E)$ has a vertex cover of size k: a set of vertices touching all edges.
    6. **Tiling Problem:** Determine if a given rectangular region can be tiled (without gaps or overlaps) using a specified set of shapes (e.g., dominoes or tetrominoes).
    7. **3D Packing Problem:** Determine if a set of 3D rectangular objects can fit into a container of fixed dimensions without overlapping, possibly rotating the objects as necessary.
    8. **Exact Cover Problem:** Given a universe U and a collection of subsets, determine if there exists a sub-collection of these subsets that covers each element of U exactly once.
    9. **Cryptarithm Solver:** Given a cryptarithm (e.g., `SEND + MORE = MONEY`), assign a unique digit to each letter so that the resulting arithmetic equation holds true.
    10. **Boolean Formula Synthesis:** Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, construct a Boolean formula in a form of a parse tree with k nodes (logic connectives and variables) that computes f.
    11. **Boolean Circuit Synthesis:** Given a Boolean function $f : \{0,1\}^n \to \{0,1\}^m$, construct a Boolean circuit with k logic gates that computes f.
    12. **Logical Equivalence Check:** Determine if two given Boolean circuits are equivalent (i.e., they compute the same Boolean function).
    13. **Scheduling Problem:** Assign n tasks to m time slots and k processors. Each task should be scheduled exactly once, precedence constraints must be satisfied, tasks sharing a resource cannot overlap, and tasks requiring multiple time slots must be scheduled contiguously.
    14. **Pancake Sorting:** Given a stack of pancakes of varying sizes, determine a sequence of flips (each flip reverses the order of the top portion of the stack) to sort the stack with the largest pancake at the bottom.
    15. **Latin Square:** Determine if a partially filled $n \times n$ grid can be completed so that each row and column contains each of n distinct symbols exactly once.
    16. **Bin Packing Problem:** Given a set of items with sizes and a fixed number of bins with given capacities, determine if all items can be placed into the bins without exceeding any bin's capacity.
    17. **Betweenness Problem:** Given a set of elements and constraints of the form (a, b, c), meaning that in any acceptable linear ordering of these elements, b must lie between a and c, determine if there exists such an ordering that satisfies all betweenness constraints.

[2] Collaborate with your classmates to cover distinct problems. Try to select problems from different domains.

Guidelines for the reduction:

- Define logical variables to represent key properties of the problem (e.g., whether a vertex is assigned a specific color, whether an item is placed in a particular bin, etc.).
- Formulate constraints that enforce the rules of the problem in propositional logic.
- Show how a solution to the SAT instance corresponds to a solution of the original problem.
- Verify that your reduction captures all valid solutions of the original problem.

Example solution for the Graph Coloring problem:

1. Define variables $x_{v,c}$ for each vertex $v \in V$ and color $c \in \{1, \ldots, k\}$, where $x_{v,c} = 1$ if vertex v is assigned color c.
2. Add constraints ensuring each vertex is assigned exactly one color:

$$\bigvee_{c=1}^{k} x_{v,c} \quad \text{for all } v \in V$$

$$\neg(x_{v,c} \wedge x_{v,c'}) \quad \text{for all } v \in V,\ c \neq c'$$

3. Add constraints ensuring no two adjacent vertices share the same color:

$$\neg(x_{u,c} \wedge x_{v,c}) \quad \text{for all } (u,v) \in E,\ c \in \{1, \ldots, k\}$$

4. Optionally, fix a specific vertex and color to reduce symmetries:

$$x_{1,1} = 1$$

5. Possible extensions and variations:
   - Bounded coloring: Require each color to be used at least $t_{\min}$ and at most $t_{\max}$ times.
   - Exact coloring: Ensure every pair of colors appears on exactly one pair of adjacent vertices.

Example solution for the Knapsack problem:

1. Define variables x_i for each item i, where $x_i = 1$ if item i is included.
2. Add constraints to ensure the total weight does not exceed the limit W:

$$\sum_i w_i x_i \leq W$$

3. Formulate the objective (though the SAT is a decision problem, you can encode the optimization (e.g., maximization) problem as a series of checks for a certain value threshold):

$$\sum_i v_i x_i \geq V_{\text{target}}$$

4. Possible extensions and variations:
   - Fractional knapsack: Allow items to be broken into smaller pieces, so that a fraction of an item (with non-proportionally less value) can be included in the knapsack.
   - Multiple knapsacks: Consider multiple knapsacks with different weight limits.