Introduction to Logic
Assignment 5 (Part B)

For this assignment, you are to create a Python program that solves one of the following puzzles by formulating the puzzle as a SAT problem and using a SAT solver to solve it. You are to choose at least one of the three puzzles given in this assignment to work on.

These are typical steps in solving a puzzle using a SAT solver.

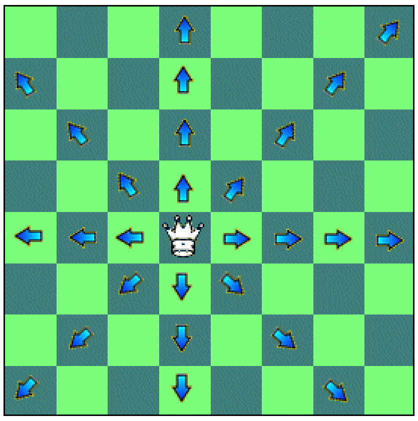
1. Define the propositional letters to be used and their meanings.
2. From the description of the puzzle, write a formula using the defined propositional letters such that the formula is satisfiable if and only the puzzle has a solution.
3. If the formula is satisfiable, extract a solution to the puzzle from a truth assignment that satisfies the formula. If the formula is unsatisfiable, report that the puzzle has no solutions.

You can use either an efficient SAT solver available on the Internet or the following SAT solver which is a simple implementation of the DPLL algorithm.

```python
# Check whether 'clause' contains complementary literals
def tautology(clause):
    for lit in clause:
        neg_lit = '~'+lit if lit[0]!='~' else lit[1:]
        if neg_lit in clause:
            return True
    return False

# Apply simple DPLL algorithm to check satisfiability
def sat(cnf):
    # Empty clause exists means unsatisfiable
    if [] in cnf:
        return None

    # Remove tautologies
    new_cnf = [c for c in cnf if not tautology(c)]

    # Empty set of clauses is obviously satisfiable
    if len(new_cnf) == 0:
        return {}

    # Select a propositional letter
    lit = new_cnf[0][0]
    p = lit if lit[0]!='~' else lit[1:]
    # Try p true
    new_cnf_p = [c for c in new_cnf if p not in c]
    new_cnf_p = [[l for l in c if l!='~'+p] for c in new_cnf_p]
    result = sat(new_cnf_p)
    if result is not None:
        result[p] = True
        return result
    else:
        # Try p false
        new_cnf_notp = [c for c in new_cnf if '~'+p not in c]
        new_cnf_notp = [[l for l in c if l!=p] for c in new_cnf_notp]
        result = sat(new_cnf_notp)
        if result is not None:
            result[p] = False
            return result
        else:
            return None

cnf1 = [['r','~s','t'],
       ['p','~r','~s'],
       ['~t'],
       ['~p','~q','t'],
       ['s','t'],
       ['~p','q']]
print(sat(cnf1))
# Output 'None' because cnf1 is unsatisfiable.

cnf2 = [['~p','t'],
        ['p','s','r'],
        ['~s', 't'],
        ['p','~q'],
        ['q','~r']]
print(sat(cnf2))
# Output {'q': True, 't': True, 'p': True} meaning that any truth assignment
# where q, t, and p true (s and r can be either true or false) satisfies
# cnf2.
```

Problem 1: n-queens

In a square chessboard, a queen is a chess piece that can move either vertically, horizontally, or diagonally in any number of blocks to attack another chess piece. In an n-queens problem, the goal is to determine whether it is possible to place n queens on an n x n chessboard such that no queen can attack any other queen. If it is possible, find which the locations on the nxn chessboard where the n queens should be placed.



http://www.learntoplaychess.com/chess-moves.htm

Input: The dimension n, where $1 \leq n \leq 9$, of the chessboard.

Output: If there is a solution, draw (using the turtle module) the n x n chessboard and the n queens; otherwise, print out "No solution".

Hint: Define the propositional letters $P_{i,j}$ where $1 \leq i,j \leq n$ which means the "a queen is placed at row i and column j of the chessboard".

Problem 2: Einstein's riddle

The Einstein's riddle is rumored to be created by Albert Einstein. There are many variations of this puzzle. One of them featuring in Life International magazine in 1962, featuring a zebra as one of the pets in the puzzle. Hence, the puzzle has also been widely known as the zebra puzzle. We will be considering a version of this puzzle.

There are five houses in a row. Each house has a different color (red, white, green, yellow, blue) and is owned by a man of a different nationality (Brit, Swede, Dane, Norwegian, German). Each of the owners has a different pet (dog, bird, cat, horse, fish), prefers a different kind of drinks (tea, coffee, milk, beer, water), and smokes a different brand of cigarettes (Pall Mall, Dunhill, Blends, Blue Master, Prince).

Furthermore, the following information is given:

1 The Brit lives in the red house.
2 The Swede keeps dogs as pets.
3 The Dane drinks tea.
4 The green house is on the right of the white house, next to each other.
5 The owner of the green house drinks coffee.
6 The person who smokes Pall Mall rears birds.
7 The owner of the yellow house smokes Dunhill.
8 The man living in the center house drinks milk.
9 The Norwegian lives in the first house.
10 The man who smokes Blends lives next to the one who keeps cats.
11 The man who keeps horses lives next to the man who smokes Dunhill.
12 The man who smokes Blue Master drinks beer.
13 The German smokes Prince.
14 The Norwegian lives next to the blue house.
15 The man who smokes Blends has a neighbour who drinks water.

The puzzle asks: Who owns the fish?

Input: None
Output: The house number and the nationality of the person who owns the fish.

Hint: Suppose the houses along the row from left to right are numbered 1 to 5 and the attributes are encoded as follows:

Attributes	red	white	green	yellow	blue
Code	01	02	03	04	05

Attributes	Brit	Swede	Dane	Norwegian	German
Code	06	07	08	09	10

Attributes	dog	bird	cat	horse	fish
Code	11	12	13	14	15

Attributes	tea	coffee	milk	beer	water
Code	16	17	18	19	20

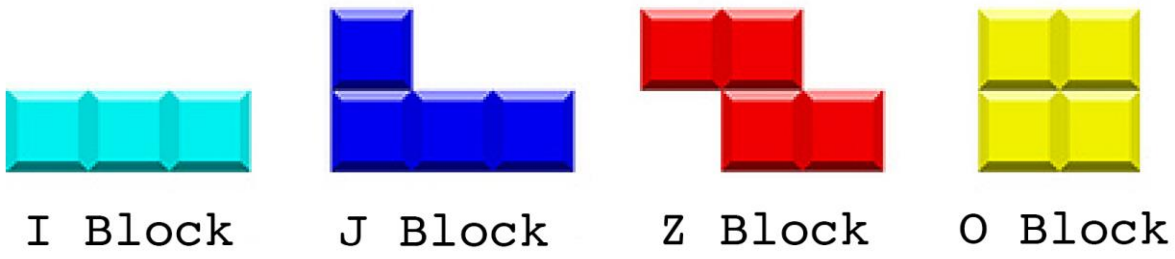
Attributes	Pall Mall	Dunhill	Blends	Blue Master	Prince
Code	21	22	23	24	25

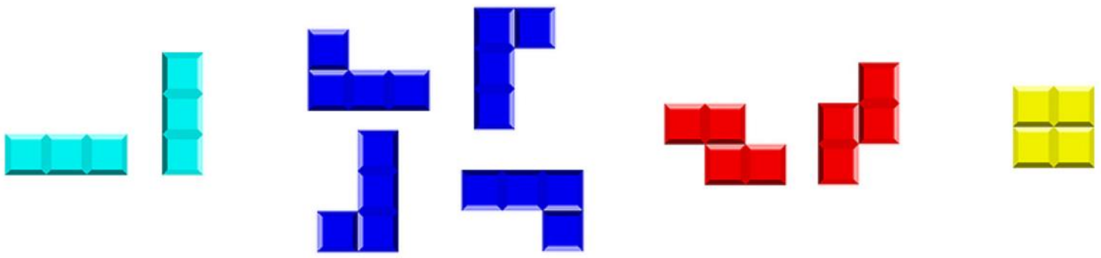
Define the propositional letters $P_{i,a}$ where i is a house number (1…5) and a is the code for an attribute (01 … 25). The meaning of $P_{i,a}$ is "The person who lives in house no. i has attribute a". For example,

- $P_{2,06}$ means "The person who lives in house no. 2 is a Brit."
- $P_{5,18}$ means "The person who lives in house no. 5 drinks milk."

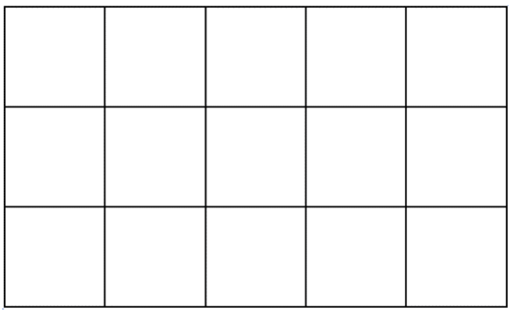
Problem 3: Tetromino

In a tetromino tiling puzzle, you are given a blank board that is divided into n x m squares and also given an unlimited supply of tetromino blocks of 4 types: I, J, Z, O, as shown in the picture below.





Blocks can be rotated in 90° units, but not flipped.



Example of a 3 x 5 board

Given the dimensions n (the number of rows) and m (the number of columns) of the board, the problem is to determine whether it is possible to fill tetromino blocks into the board so the blocks fit the board perfectly, i.e. blocks do not overlap and there is no unfilled space left on the board.

Input: The dimensions n and m of the board, where 1 ≤ n,m ≤ 9
Output: If there is a solution, draw (using the turtle module) an n x m board perfectly filled with tetromino blocks; otherwise, print out "No solution".