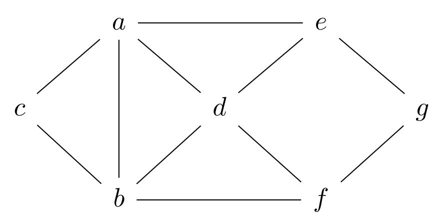
Question 1. Apply backtracking to solve problem of finding a Hamiltonian Circuit in the following graph (using Brute-force with alphabetic tie-breaking):



Question 2. In the lecture you were provided with the following recursive algorithm for solving the n-queens problem:

```
def rec_nQueens(size, queens):
    """
    Recursively computes a solution for the n-Queens puzzle.

    param queens: The currently placed queens, e.g. [4,2] represents
    that on row 0 we placed a queen on the 4th position, and row 1 we placed
    a queen on the 2nd position.
    type queens: List

    param size: The size of the puzzle
    type size: Non-negative integer.
    """

    if size <= len(queens):
        return queens

    for col in range(size):
        if constraint(queens, col):
            queens.append(col)
            candidate_sol = rec_nQueens(size, queens)

            if candidate_sol: # In Python any non-empty list is True, e.g. the base case.
                return candidate_sol
            queens.pop()
    return False
```

Currently the algorithm only provides a single solution. Augment the function so that it returns every solution to the problem (you do not need to provide the `constraint()` method).

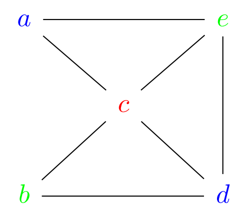
Question 3. The *Mountain Goats* from last week (PS5, Q4) describes a problem that can be solved using exhaustive search, design an algorithm in pseudo code that uses backtracking to solve the problem.

Question 4. *M-colour Problem*

The m-colour problem is another famous graph-problem. Given a graph G, and some number m, the problem is to determine whether or not there is a way colour the vertices of G using m colours in such a way that no neighbouring vertices share the same colour.

Design a pseudocode algorithm which uses backtracking to solve the m-colour problem.

An example of a solution for m=3



Question 5.

Given a maze that is traversed orthogonally (horizontally and vertically) with a Start and End, design an back-tracking algorithm in pseudocode that finds a way out, or returns false if there is none.

```
XXXXXXXXXXXX
X     X    X
X X X XXX XX
XXX X X    X
X   X   X XX
X XXXXXXX XX
X      X   X
XXXX X X X X
S  XXX X X X
XX     X X X
X  X X X X X
XXXXXXXXEXXX
```