ASSIGNMENT.

Title : Constraint satisfaction problem.

Problem statement:
Implement crypt-arithmetic problem or n-queens or graph-coloring problem (Branch-and-Bound and Backtracking)

Objectives:
- To learn and implement constraint satisfaction problem

Outcome:
We will be able to: implement constraint satisfaction problem

Software and Hardware requirements:
- Operating System: 64-bit Open sour Linux or its derivative.
- Programming language : Python / Java.

Theory:
Constraint satisfaction problem:

A constraint satisfaction problem (CSP) consists of
- a set of variables
- a domain for each variable, and
- a set of constraints

The aim is to choose a value for each variable so that resulting possible world satisfies the constraints.

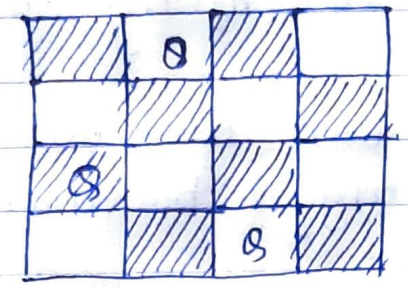
- A finite CSP has a finite set of variables and a finite domain for each variable.
- Given a CSP, there are a number of tasks that can be performed:
  - Determine whether or not there is a model.
  - Find a model
  - Find all of the models or enumerate the models.
  - Count the number of models.
  - Find the best model, given a measure of how good models are.
  - Determine whether some statement holds in all models

Backtracking.

- It is an algorithmic technique for solving problems recursively by trying to build a solution incrementally one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time.
- Three problems in backtracking are:
  1. Decision problem : In this, we search for a feasible solution.
  2. Optimization problem : In this, we search for best solution.
  3. Enumeration problem: In this, we find all feasible solution.

N- Queens Problem

- It is the problem of placing N queens on an N×N chessboard so that no two queens attack each other.
- For example, following is the solution for 4 queen problem.



- The expected output is a binary matrix which has 1's for the blocks where queens are placed. For example, following is the output

```
{ 0, 1, 0, 0}
{ 0, 0, 0, 1}
{ 1, 0, 0, 0}
{ 0, 0, 1, 0}
```

Backtracking algorithm:

1) Start in the leftmost column
2) If all queens are placed
   return true
3) Try all rows in the current column Do following for every tried row.
   a) If queen can be placed safely in this row then mark this [row, column] as a part of this solution and recursively check if placing queen here

leads to a solution.

b) If placing the queen in [row, column] leads to a solution then return true.

c) If placing queen doesn't lead to a solution then unmark this [row, column] (Backtrack) and go to step (a). to try other rows

4) If all rows have been tried but nothing worked, return false to trigger backtracking.

Test cases and analysis:

Sr.no.	No. of queens	Output.
1.	5 [success]	[1, 0, 0, 0, 0] [0, 0, 1, 0, 0] [0, 0, 0, 0, 1] [0, 1, 0, 0, 0] [0, 0, 0, 1, 0]
2.	4. [success]	[0, 1, 0, 0] [0, 0, 0, 1] [1, 0, 0, 0] [0, 0, 1, 0]
3.	8. [success]	[1, 0, 0, 0, 0, 0, 0, 0] [0, 0, 0, 0, 1, 0, 0, 0] [0, 0, 0, 0, 0, 0, 0, 1] [0, 0, 0, 0, 0, 1, 0, 0] [0, 0, 1, 0, 0, 0, 0, 0] [0, 0, 0, 0, 0, 0, 1, 0] [0, 1, 0, 0, 0, 0, 0, 0] [0, 0, 0, 1, 0, 0, 0, 0]

Conclusion:

Thus, we have successfully implemented a constraint satisfaction problem (i.e. CSP) in n-queens problem using backtracking.