Assignment 12

Perumalla Dharan
AP21110010201

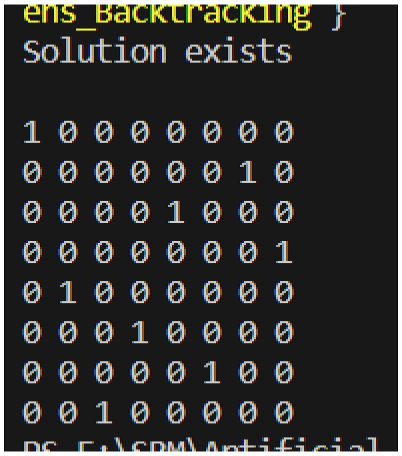
1.

Backtracking

```
#include <iostream>
using namespace std;

int board[8][8] = {0};

bool isSafe(int row, int col)
{
    for (int i = 0; i < col; i++)
    {
        if (board[row][i] == 1)
            return false;
    }

    for (int i = row, j = col; i >= 0 && j >= 0; i--,
j--)
    {
        if (board[i][j] == 1)
            return false;
    }

    for (int i = row, j = col; i < 8 && j >= 0; i++,
j--)
    {
        if (board[i][j] == 1)
            return false;
```

```cpp
    }

    return true;
}

bool solve(int col)
{
    if (col >= 8)
        return true;

    for (int i = 0; i < 8; i++)
    {
        if (isSafe(i, col))
        {
            board[i][col] = 1;

            if (solve(col + 1))
                return true;

            board[i][col] = 0;
        }
    }

    return false;
}

int main()
{
    if (solve(0))
    {
        cout << "Solution exists\n";
        for (int i = 0; i < 8; i++)
        {
            cout << endl;
            for (int j = 0; j < 8; j++)
            {
                cout << board[i][j] << " ";
            }
        }
    }
    else
    {
        cout << "Solution does not exist\n";
    }

    return 0;
}
```

Output



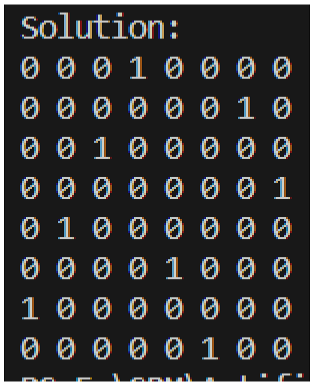
```
ens_Backtracking }
Solution exists

1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
PS E:\SRM\Artificial
```

2.

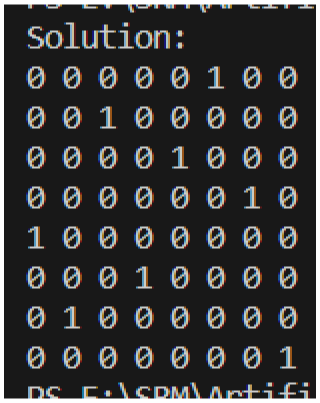
CSP

```
from constraint import *
import random

def solve_8_queens_csp():
    problem = Problem()
    cols = range(8)
    rows = range(8)
    problem.addVariables(cols, rows)
    for col1 in cols:
        for col2 in cols:
            if col1 < col2:
                problem.addConstraint(lambda row1, row2,
col1=col1, col2=col2:
                                      row1 != row2 and
row1 + col1 != row2 + col2 and row1 - col1 != row2 -
col2,
                                      (col1, col2))

    solutions = problem.getSolutions()
    if len(solutions) > 0:
        print("Solution:")
        solution = random.choice(solutions)
        board = [[0 for _ in range(8)] for _ in
range(8)]
        for col, row in solution.items():
            board[row][col] = 1
        for row in board:
            print(" ".join(map(str, row)))
    else:
        print("No solution exists.")

if __name__ == "__main__":
    solve_8_queens_csp()
```

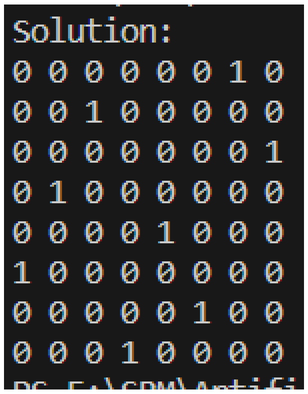
Output



```
Solution:
0 0 0 1 0 0 0 0
0 0 0 0 0 0 1 0
0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0
1 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0
PS E:\SRM\Artifi
```



```
Solution:
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 1 0
1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 1
PS E:\SRM\Artifi
```



```
Solution:
0 0 0 0 0 0 1 0
0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0
1 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0
0 0 0 1 0 0 0 0
PS E:\SRM\Artifi
```