In [5]:
```
class NQueensProblem:
    def __init__(self, n):
        self.queens = [0] * n
        self.numSolutions = 0

    def solve(self):
        self.solve_helper(0)

    def solve_helper(self, row):
        if row == len(self.queens):
            self.numSolutions += 1
            self.print_solution()
        else:
            for col in range(len(self.queens)):
                self.queens[row] = col
                if self.is_valid(row, col):
                    self.solve_helper(row + 1)

    def is_valid(self, row, col):
        for i in range(row):
            diff = abs(self.queens[i] - col)
            if diff == 0 or diff == row - i:
                return False
        return True

    def print_solution(self):
        if self.numSolutions == 1:
            print("Solution: ", end="")
            for i in range(len(self.queens)):
                print(self.queens[i], end=" ")
            print()
            print("The Matrix Representation:")
            arr = [[0] * len(self.queens) for _ in range(len(self.qu
            for i in range(len(self.queens)):
                for j in range(len(self.queens)):
                    if j == self.queens[i]:
                        arr[i][j] = 1
            for i in range(len(self.queens)):
                for j in range(len(self.queens)):
                    print(arr[i][j], end=" ")
                print()

if __name__ == "__main__":
    n = int(input("Enter N Queens Problem: "))
    NQueensProblem = NQueensProblem(n)
    NQueensProblem.solve()
```

```
Enter N Queens Problem: 9
Solution: 0 2 5 7 1 3 8 6 4
The Matrix Representation:
1 0 0 0 0 0 0 0 0
0 0 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 1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 1 0 0
0 0 0 0 1 0 0 0 0
```