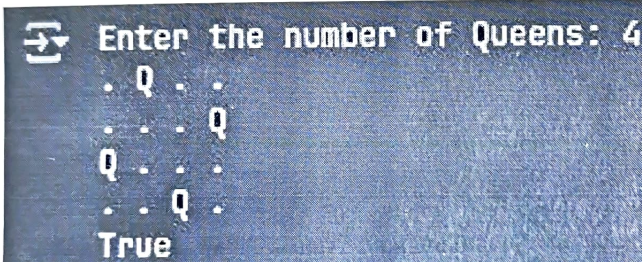


PROGRAM

```
def solve(queens, N):  
    if len(queens) == N:  
        print("\n", join(" ".join("Q" if i == c else "." for i in range(N))  
                           for c in queens))  
        return True  
    for col in range(N):  
        if all(col != c and abs(len(queens) - r) != abs(col - c)  
               for r, c in enumerate(queens)):  
            if solve(queens + [col], N):  
                return True  
    return False  
  
N = int(input("Enter the number of Queens: "))  
solve([], N)
```

GOOGLE COLLAB



```
Enter the number of Queens: 4  
. Q . .  
. . . Q  
Q . . .  
. . Q .  
True
```

OUTPUT

```
Enter the number of Queens: 4  
. Q . .  
. . . Q  
Q . . .  
. . Q .  
True
```

10/09/2024

EXP NO: 1EXP NAME: N QUEENS ALGORITHMAIM: To implement the N Queens algorithm using python.ALGORITHM:

- 1) Get number of queens from user as input
- 2) Pass the number value into the solve function.
- 3) Check if the board is filled with queens
- 4) If true print the board
- 5) Else loop col in range of N.
- 6) Inside the loop check all (.) does the following checks
- 7) The for r, c in $\text{enumerate}(\text{board})$ gives us the index r and value c .
- 8) Next we check if $\text{col} \neq c$ (column not same as current)
- 9) And the diagonal $\text{abs}(\text{len}(\text{queens}) - r) \neq \text{abs}(\text{col} - c)$:
- 10) Where
 - $\text{col} \rightarrow$ The new column where we are trying to place queen
 - $c \rightarrow$ The column where previous queen is placed
 - $\text{len}(\text{queens}) \rightarrow$ The number of queens placed (next row)
 - $r \rightarrow$ The row of previous queen.
- 11) If true we pass the next queen value along with new column value.

RESULT:

Thus the program has been successfully implemented in python and output has been implemented.