NAME:	Harshal Chawan
UID:	2021300019
SUBJECT	Design and Analysis of Algorithm
EXPERIMENT NO :	07
DATE OF PERFORMANCE	10/04/2023
DATE OF SUBMISSION	17/04/2023
AIM:	To use backtracking algorithm to solve N queens problem.
PROBLEM STATEMENT 1:	**N Queen's problem.**

ALGORITHM and THEORY:	
	```
	function solveNQueens(board, col, n):

	  if col >= n:
	    print board
	    return true
	  for row from 0 to n-1:
	    if isSafe(board, row, col, n):
	      board[row][col] = 1
	      if solveNQueens(board, col+1, n):
	        return true
	      board[row][col] = 0
	  return false

	function isSafe(board, row, col, n):
	  for i from 0 to col-1:
	    if board[row][i] == 1:
	      return false
	```

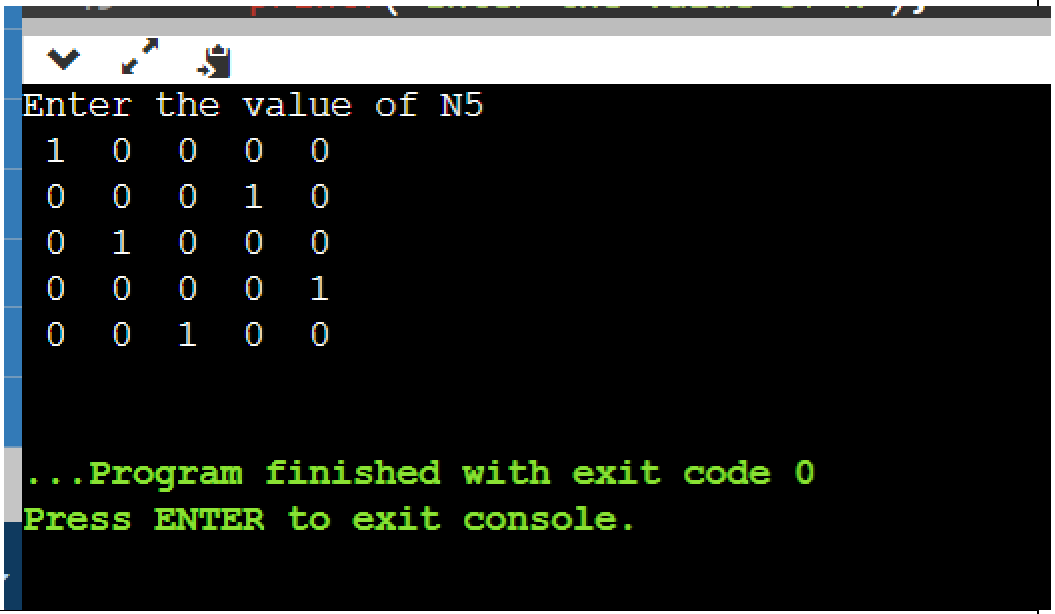
```
   for i,j from row-1, col-1 to 0, 0 by -
     1:if board[i][j] == 1:
       return false
   for i,j from row+1, col-1 to n-1, 0 by 1, -
     1:if board[i][j] == 1:
       return
   falsereturn
   true

 board = empty NxN
 chessboard
 solveNQueens(board, 0, N)
```

Program:

```
#include <stdbool.h>
#include <stdio.h>
int N;
void printSolution(int board[N][N])
{
        for (int i = 0; i < N; i++) {
                for (int j = 0; j < N; j++)
                        printf(" %d ", board[i][j]);
                printf("\n");
        }
}

bool isSafe(int board[N][N], int row, int col)
{
        int i, j;

        for (i = 0; i < col; i++)
                if (board[row][i])
                        return false;

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

        for (i = row, j = col; j >= 0 && i < N; i++, j--)
                if (board[i][j])
                        return false;

        return true;
```

```
}
bool solveNQUtil(int board[N][N], int col)
{
        if (col >= N)
                return true;
        for (int i = 0; i < N; i++) {
                if (isSafe(board, i, col)) {
                        board[i][col] = 1;
                        if (solveNQUtil(board, col + 1))
                                return true;
                        board[i][col] = 0;
                }
        }
        return false;
}


bool solveNQ()
{
   printf("Enter the value of N");
   scanf("%d",&N);
   int board[N][N];
   for(int i=0; i<N; i++)
   {
      for(int j=0; j<N; j++)
         {
            board[i][j]=0;
         }
   }

        if (solveNQUtil(board, 0) == false) {
                printf("Solution does not exist");
                return false;
        }

        printSolution(board);
        return true;
}

int main()
{
        solveNQ();
        return 0;
```

	}

OUTPUT:	
	```
	Enter the value of N5
	 1  0  0  0  0
	 0  0  0  1  0
	 0  1  0  0  0
	 0  0  0  0  1
	 0  0  1  0  0


	...Program finished with exit code 0
	Press ENTER to exit console.
	```

CONCLUSION:	By performing the above experiment I was able to implement the N queens problem to print the chess board solution with 8queens not attacking each other.