NAME:	Shubham Vishwakarma
UID No.	2021700071
BRANCH:	S.Y CSE-DS
BATCH:	D
SUBJECT	Design and Analysis of Algorithms
EXPERIMENT No.	7
Date of Performance	10/04/2023
Date of Submission	14/04/2023

AIM:	**Backtracking (To implement N Queens problem using backtracking.)**
Program 1	
PROBLEM STATEMENT :	Implement the N queen problem for 4x4 chess board.
ALGORITHM/ THEORY:	**Step 1 -** Place the queen row-wise, starting from the left-most cell. **Step 2 -** If all queens are placed then return true and print the solution matrix. **Step 3 -** Else try all columns in the current row. • **Condition 1 -** Check if the queen can be placed safely in this column then mark the current cell [Row, Column] in the solution matrix as 1 and try to check the rest of the problem recursively by placing the queen here leads to a solution or not. • **Condition 2 -** If placing the queen [Row, Column] can lead to the solution return true and print the solution for each queen's position. • **Condition 3 -** If placing the queen cannot lead to the solution then unmark this [row, column] in the solution matrix as 0, BACKTRACK, and go back to condition 1 to try other rows. **Step 4 -** If all the rows have been tried and nothing worked, return false to trigger backtracking.

PROGRAM:

```c
#include<stdio.h>
int board[100][100];

int safe(int r,int c,int n){
    for(int i=0;i<r;i++){
        if(board[i][c] == 1)
          return 0;
    }
    for(int i=r,j=c;i>=0 && j>=0;i--,j--){
       if(board[i][j] == 1)
         return 0;
    }
    for(int i=r,j=c;i>=0 && j<n;i--,j++){
       if(board[i][j] == 1)
         return 0;
    }
    return 1;
}

void printsol(int n){
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
           if(board[i][j]==1){
             printf("Q ");
           }
           else
             printf("_ ");
        }
        printf("\n");
    }
    printf("\n");
}
void position(int n,int r){
    if(r>=n){
      printsol(n);
      return;
    }
    for(int i=0;i<n;i++){
        if(safe(r,i,n)){
           board[r][i] = 1;
           position(n,r+1);
           board[r][i] = 0;
        }
    }
}

int main(){
```

```c
    int n;
    printf("\nEnter the size of the board: ");
    scanf("%d",&n);
    printf("\nThe possible outputs are: \n");
    position(n,0);
}
```

RESULT:

```
PS C:\Users\smsha\Desktop\SEM 4\DAA\Practicals\Exp7\output> & .\'Nqueen.exe'

Enter the size of the board: 5

The possible outputs are:
Q _ _ _ _
_ _ Q _ _
_ _ _ _ Q
_ Q _ _ _
_ _ _ Q _

Q _ _ _ _
_ _ _ Q _
_ Q _ _ _
_ _ _ _ Q
_ _ Q _ _

_ Q _ _ _
_ _ _ Q _
Q _ _ _ _
_ _ Q _ _
_ _ _ _ Q

_ Q _ _ _
_ _ _ _ Q
_ _ Q _ _
Q _ _ _ _
_ _ _ Q _

_ _ Q _ _
Q _ _ _ _
_ _ _ Q _
_ Q _ _ _
_ _ _ _ Q
```

CONCLUSION :

- N Queen problem is a classical puzzle that beautifully develops the concept of *Backtracking*.
- The time complexity of the brute force backtracking algorithm is O(N×N!)O(N×N!). However, using *bitmasking*

	the time complexity can be optimized to O(N!)O(N!). • The space complexity irrespective of the approach is O(N2)O(N2) because we need to print a 2-dimensional array as the answer.