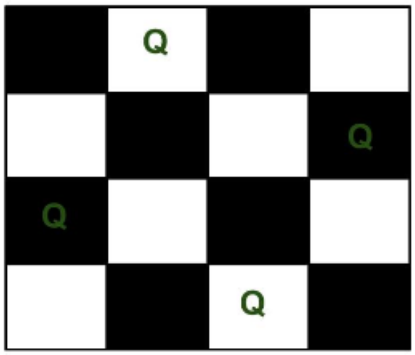
| | |
|---|---|
| **NAME:** | Deepanshu Aggarwal |
| **UID:** | 2021300002 |
| **BRANCH:** | Computer engineering |
| **BATCH:** | A |
| **SUBJECT:** | DAA |
| **EXPT NO:** | 7 |
| **AIM:** | Implementing N queens using backtrack strategy. |
| **THEORY:** | Backtracking is an algorithmic technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time (by time, here, is referred to the time elapsed till reaching any level of the search tree).<br><br>The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other. For example, the following is a solution for the 4 Queen problem.<br><br>  <br><br><br>The expected output is in form of a matrix that has ‘Q’s for the blocks where queens are placed and the empty spaces are represented by ‘.’s . |

**PROGRAM:**

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int board[20], count;

int main()
{
    int n, i, j;
    void queen(int row, int n);

    printf(" - N Queens Problem Using Backtracking -");
    printf("\n\nEnter number of Queens:");
    scanf("%d", &n);
    queen(1, n);
    return 0;
}

void print(int n)
{
    int i, j;
    printf("\n\nSolution %d:\n\n", ++count);

    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= n; ++j)
        {
            if (board[i] == j)
                printf(" Q");
            else
                printf(" .");
        }
        printf("\n");
    }
}

int place(int row, int column)
{
    int i;
    for (i = 1; i <= row - 1; ++i)
    {
        if (board[i] == column)
            return 0;
        else if (abs(board[i] - column) == abs(i - row))
```

```
            return 0;
    }

    return 1;
}

void queen(int row, int n)
{
    if (count == 5)
        return;
    int column;
    for (column = 1; column <= n; ++column)
    {
        if (place(row, column))
        {
            board[row] = column;
            if (row == n)
                print(n);
            else
                queen(row + 1, n);
        }
    }
}
```

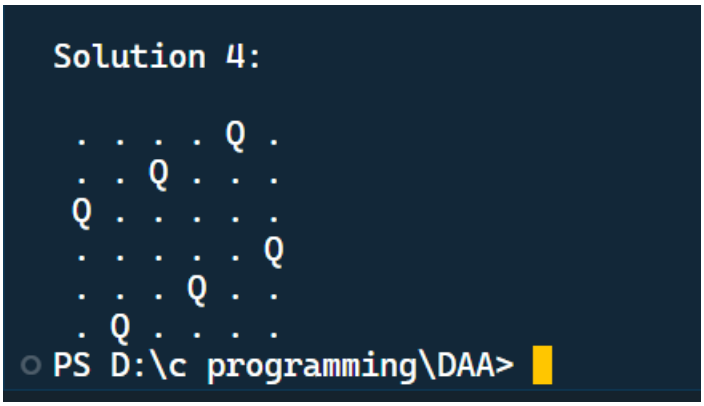
**OUTPUT:**

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    SQL CONS

PS D:\c programming> cd "d:\c programming\DAA\"
 - N Queens Problem Using Backtracking -

Enter number of Queens:8


Solution 1:

 Q . . . . . . .
 . . . . Q . . .
 . . . . . . . Q
 . . . . . Q . .
 . . Q . . . . .
 . . . . . . Q .
 . Q . . . . . .
 . . . Q . . . .


Solution 2:

 Q . . . . . . .
 . . . . . Q . .
 . . . . . . . Q
 . . Q . . . . .
 . . . . . . Q .
 . . . Q . . . .
 . Q . . . . . .
 . . . . Q . . .
```

```
Solution 3:

 Q . . . . . . .
 . . . . . . Q .
 . . . Q . . . .
 . . . . . Q . .
 . . . . . . . Q
 . Q . . . . . .
 . . . . Q . . .
 . . Q . . . . .


Solution 4:

 Q . . . . . . .
 . . . . . . Q .
 . . . . Q . . .
 . . . . . . . Q
 . Q . . . . . .
 . . . Q . . . .
 . . . . . Q . .
 . . Q . . . . .


Solution 5:

 . Q . . . . . .
 . . . Q . . . .
 . . . . . Q . .
 . . . . . . . Q
 . . Q . . . . .
 Q . . . . . . .
 . . . . . . Q .
 . . . . Q . . .
PS D:\c programming\DAA>
```

## SOLUTION – 2

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    SQL CONSOLE

PS D:\c programming> cd "d:\c programming\DAA\" ;
 - N Queens Problem Using Backtracking -

Enter number of Queens:6


Solution 1:

 . Q . . . .
 . . . Q . .
 . . . . . Q
 Q . . . . .
 . . Q . . .
 . . . . Q .


Solution 2:

 . . Q . . .
 . . . . . Q
 . Q . . . .
 . . . . Q .
 Q . . . . .
 . . . Q . .


Solution 3:

 . . . Q . .
 Q . . . . .
 . . . . Q .
 . Q . . . .
 . . . . . Q
 . . Q . . .
```

| | |
|---|---|
| |  |
| **CONCLUSION:** | Through this experiment, I understood and learnt the concept of backtracking. Also I implement it in solving the n-queen problem. |