

## Experiment Number: 7

**Problem Statement:-** Determine the positions of 8 different Queens on a Single Chess Board.

**Aim:-** Implementation of 8 Queens Problem.

### Theory:-

A classic combinatorial problem is to place eight queens on an 8 X 8 chessboard so that no two "attack", that is so that no two of them are on the same row, column or diagonal. The rows and columns of the chessboard be numbered 1 through 8. The queens may also be numbered 1 through 8. Since each queen must be on a different row, without loss of generality queen  $i$  is to be placed on row  $i$ . All solutions to the 8 queens problem can therefore be represented as 8-tuples  $(x_1, \dots, x_8)$  where  $x_i$  is the column on which queen  $i$  is placed.

If the squares of the chessboard being numbered as the indices of the two dimensional array  $A(l:n, l:n)$  then for every element on the same diagonal which runs from the upper left to the lower right, each element has the same "row - column" value. Also, every element on the same diagonal which goes from the upper right to the lower left has the same "row + column" value. Suppose two queens are placed at positions  $(i, j)$  and  $(k, l)$ . Then by the above they are on the same diagonal only if

$$i - j = k - l \text{ or } i + j = k + l$$

The first equation implies

$$j - l = i - k$$

while the second implies

$$j - l = k - i$$

Therefore two queens lie on the same diagonal if and only if  $|j - l| = |i - k|$

### Algorithm: -

Algorithm Place(k,i)

// Returns true if a queen can be placed in  $k^{\text{th}}$  row and  $i^{\text{th}}$  column. Otherwise it returns false. //  $x[]$  is a global array whose first  $(k-1)$  values have been set. Abs(r) returns the absolute //value of r.

```
{
for j= 1 to (k-1) do
if ((X(j) = i) //two in the same column
or (Abs(X(j) - i) = Abs(j - k))) //in the same diagonal//
then return (false)
return( true)
}
```

*Algorithm 1. Can a new queen be placed?*

**Algorithm NQUEENS(k,n)**

// using backtracking this procedure prints all possible placements of  $n$  queens on an  $n \times n$  // chessboard so that they are nonattacking.

```
{
    for i=1 to n do
    {
```

```

    if Place(k,i) then
        {
x[k]=i;
        If (k=n) then write (x[1:n]);
elseNQueens(k+1,n);
        }
    }
}

```

*Algorithm 2: All solutions to the n-queens problem.*

Algorithm Place(k,i) returns a boolean value which is true if the kth queen can be placed in column i. It tests both if i is distinct from all previous values  $x(1), \dots, x(k-1)$  and also if there is no other queen on the same diagonal. Its computing time is  $O(k-1)$ .

AlgoritmNQueens(1,n) gives the precise solution to the n-queens problem using algorithm Place.

Following figure shows one of the possible solution to 8-Queen problem.

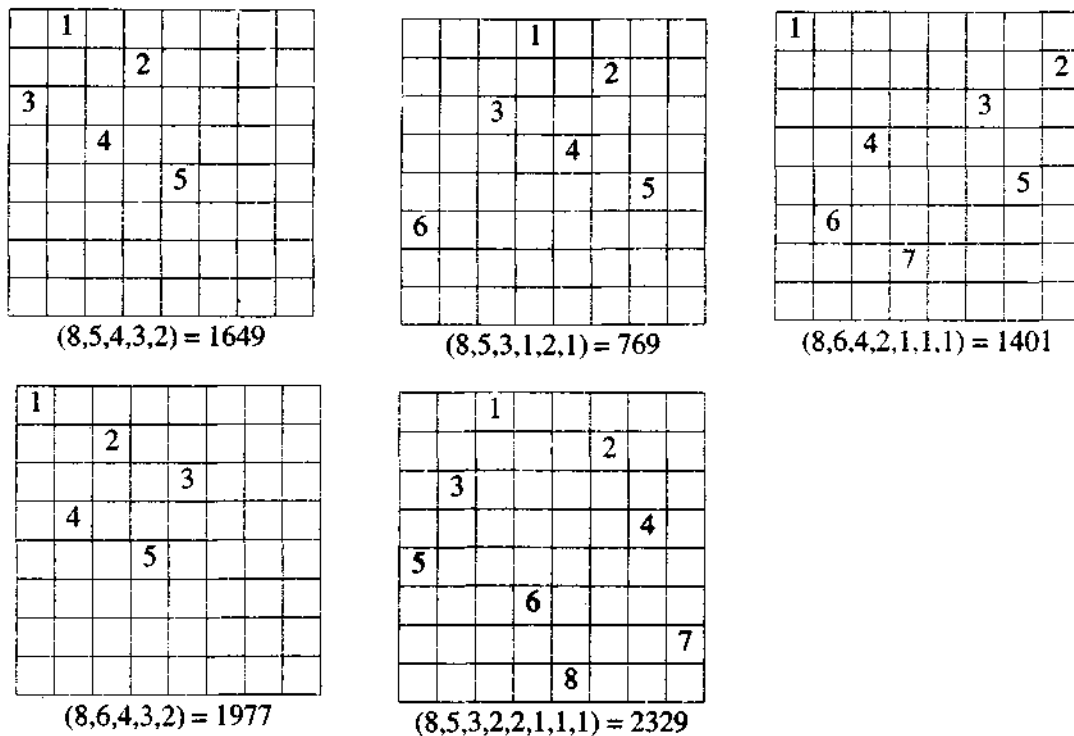


Figure 1. Five walks through the 8-queens problem plus estimates of the tree size

## Questionnaires

<b>Ques.1</b>	What is backtracking?
<b>Ans</b>	Backtracking is an algorithm design technique for finding all solutions to constraint satisfaction computational problems, that incrementally builds candidates to the solutions, and abandons each partial candidate c as soon as it determines that c cannot possibly be completed to a valid solution.
<b>Ques.2</b>	Define explicit constraints.
<b>Ans</b>	Explicit constraints are rules that restrict each input to take the values only from a given set.
<b>Ques.3</b>	Define implicit constraints.
<b>Ans</b>	Implicit constraints are rules that determine which of the tuples in the solution space satisfy the criterion function.
<b>Ques.4</b>	What is 8 Queen's Problem?
<b>Ans</b>	It's a classic combinatorial problem to place 8 queens on an 8 X 8 chessboard in such a way that any two queens should not attack, i.e. any two queens should not be on the same row, column or diagonal.

## Program

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void nQueens(int r,int n,int column[]);
int place(int r,int c,int column[]);
void main()
{
    int n;
    int column[50];
    clrscr();
    printf("Enter n:");
    scanf("%d",&n);
    nQueens(1,n,column);
    getch();
}
void nQueens(int r,int n,int column[])
{
    int i,c;
    for(c=1;c<=n;c++)
        if(place(r,c,column))
        {
```

```

column[r]=c;
if(r==n)
{
for(i=1;i<=n;i++)
printf("Queen no %d is placed in row no %d and column no %d\n",i,i,column[i]);
exit(1);
}
else
nQueens(r+1,n,column);
}
}
int place(int r,int c,int column[])
{
int j;
for(j=1;j<=r-1;j++)
if(((column[j]==c)||((abs(column[j]-c))==abs(j-r))))
return 0;
return 1;
}

```

**OUTPUT:-**

## **EXPERIMENT N0-7**

### **AIM: Implementation of 8-QUEEN problem using backtracking**

#### **CODE:**

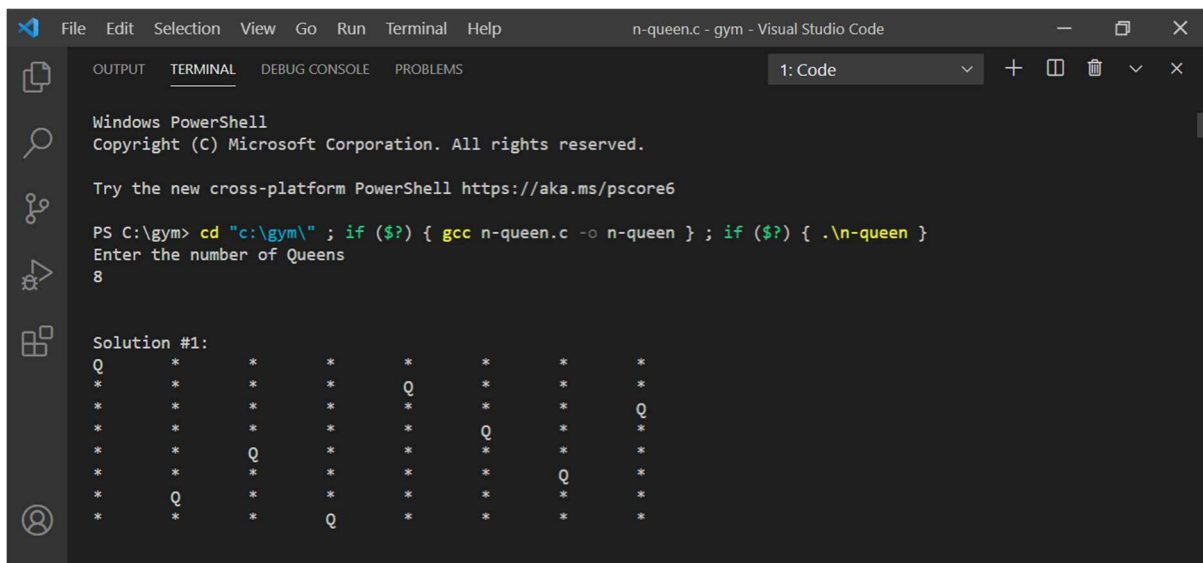
```
#include<stdio.h>
#include<conio.h>
#include<math.h>
int a[30],count=0;
int place(int pos) {
    int i;
    for (i=1;i<pos;i++) {
        if(((a[i]==a[pos])||((abs(a[i]-a[pos])==abs(i-pos)))))
            return 0;
    }
    return 1;
}
void print_sol(int n) {
    int i,j;
    count++;
    printf("\n\nSolution #0%d:\n",count);
    for (i=1;i<=n;i++) {
        for (j=1;j<=n;j++) {
            if(a[i]==j)
                printf("Q\t"); else
                printf("*\t");
        }
        printf("\n");
    }
```

```

    }
}
void queen(int n) {
    int k=1;
    a[k]=0;
    while(k!=0) {
        a[k]=a[k]+1;
        while((a[k]<=n)&&!place(k))
            a[k]++;
        if(a[k]<=n) {
            if(k==n)
                print_sol(n); else {
                    k++;
                    a[k]=0;
                }
            } else
                k--;
        }
    }
}
void main() {
    int i,n;
    printf("Enter the number of Queens\n");
    scanf("%d",&n);
    queen(n);
    printf("\nTotal solutions=%d",count);
    getch();
}

```

## OUTPUT:



```
File Edit Selection View Go Run Terminal Help n-queen.c - gym - Visual Studio Code
OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS 1: Code
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\gym> cd "c:\gym\" ; if ($?) { gcc n-queen.c -o n-queen } ; if ($?) { .\n-queen }
Enter the number of Queens
8

Solution #1:
Q      *      *      *      *      *      *      *
*      *      *      *      Q      *      *      *
*      *      *      *      *      *      *      Q
*      *      *      *      *      Q      *      *
*      *      Q      *      *      *      *      *
*      *      *      *      *      *      Q      *
*      Q      *      *      *      *      *      *
*      *      *      Q      *      *      *      *
```

## CONCLUSION:

By performing the n-queen problems we can conclude:

- In backtracking, at each level branching factor decreases by 1 and it creates a new problem of size (n-1). With n choices, it creates n different problems of size (n-1) at level 1.
- Place function determines the position of the queen in O(n) time. This function is called n times.
- Thus, the recurrence of n-Queen problem is defined as,  
 $T(n) = n * T(n-1) + n^2$ . Solution to recurrence would be O(n!).