EXPERIMENT NO- 08

AIM: Program to implement N Queen problem (8-Queen) by backtracking.

PROBLEM STATEMENT : Write a program to implement N Queen problem (8-Queen) by backtracking.

Resource Required: Pentium IV, Turbo C, Printer, Printout Stationary

THEORY:

The eight queens puzzle is the problem of putting eight chess queens on an 8×8 chessboard such that none of them is able to capture any other using the standard chess queen's moves. The queens must be placed in such a way that no two queens would be able to attack each other. Thus, a solution requires that no two queens share the same row, column, or diagonal. The eight queens puzzle is an example of the more general n queens puzzle of placing n queens on an n×n chessboard, where solutions exist only for n = 1 or n ≥ 4.
The N queen's problem solved using the recursive approach, where in the placement of the queen is decided by every recursive call to the function that decides queen's placement at every level. The main care has to be taken to place the queen is that:

1. No queen can be placed in the immediate vicinity of the queen at preceding level.
2. No queens should be on same diagonal.

The problem can be quite computationally expensive as there are 4,426,165,368 or 64! / (56 !8!)) possible arrangements of eight queens on the board, but only 92 solutions. For example, just by applying a simple rule that constrains each queen to a single column (or row), it is possible to reduce the number of possibilities to just 16,777,216 (8^8) possible combinations, which is computationally manageable for n=8, but would be intractable for problems of n=1,000,000. This heuristic solves n queens for any n, n ≥ 4 or n = 1:

Steps:

1. Divide n by 12. Remember the remainder (n is 8 for the eight queens puzzle).
2. Write a list of the even numbers from 2 to n in order.
3. If the remainder is 3 or 9, move 2 to the end of the list.
4. Append the odd numbers from 1 to n in order, but, if the remainder is 8, switch pairs (i.e. 3, 1, 7, 5, 11, 9, …).
5. If the remainder is 2, switch the places of 1 and 3, then move 5 to the end of the list.
6. If the remainder is 3 or 9, move 1 and 3 to the end of the list.
7. Place the first-column queen in the row with the first number in the list, place the second-column queen in the row with the second number in the list, etc.

For n = 8 this results in the solution shown above. A few more examples follow.

- 14 queens (remainder 2): 2, 4, 6, 8, 10, 12, 14, 3, 1, 7, 9, 11, 13, 5.
- 15 queens (remainder 3): 4, 6, 8, 10, 12, 14, 2, 5, 7, 9, 11, 13, 15, 1, 3.
- 20 queens (remainder 8): 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 3, 1, 7, 5, 11, 9, 15, 13, 19, 17.

Algorithm:

Algorithm Queen (n)
// Input: Total number of Queen‟s n for column← 1 to n do

{
if (place(row, column)) then
{
board[row]=column // no conflict so place queen if(row==n) then //dead end
print_board(n) // printing the board configuration else // try next queen with next position
Queen (row+1, n);} }
Algorithm place (row, column) // This algorithm is for placing the queen at appropriate position
// Input: row and column of the chessboard for i ← 1 to row-1 do
{

// checking for column and diadonal conflicts if (board[i] = column) then
return 0; else
if (abs(board[i] – column) = abs(i-row)) then return 0;
}
// no conflicts hence Queen can be placed return 1;

CODE:

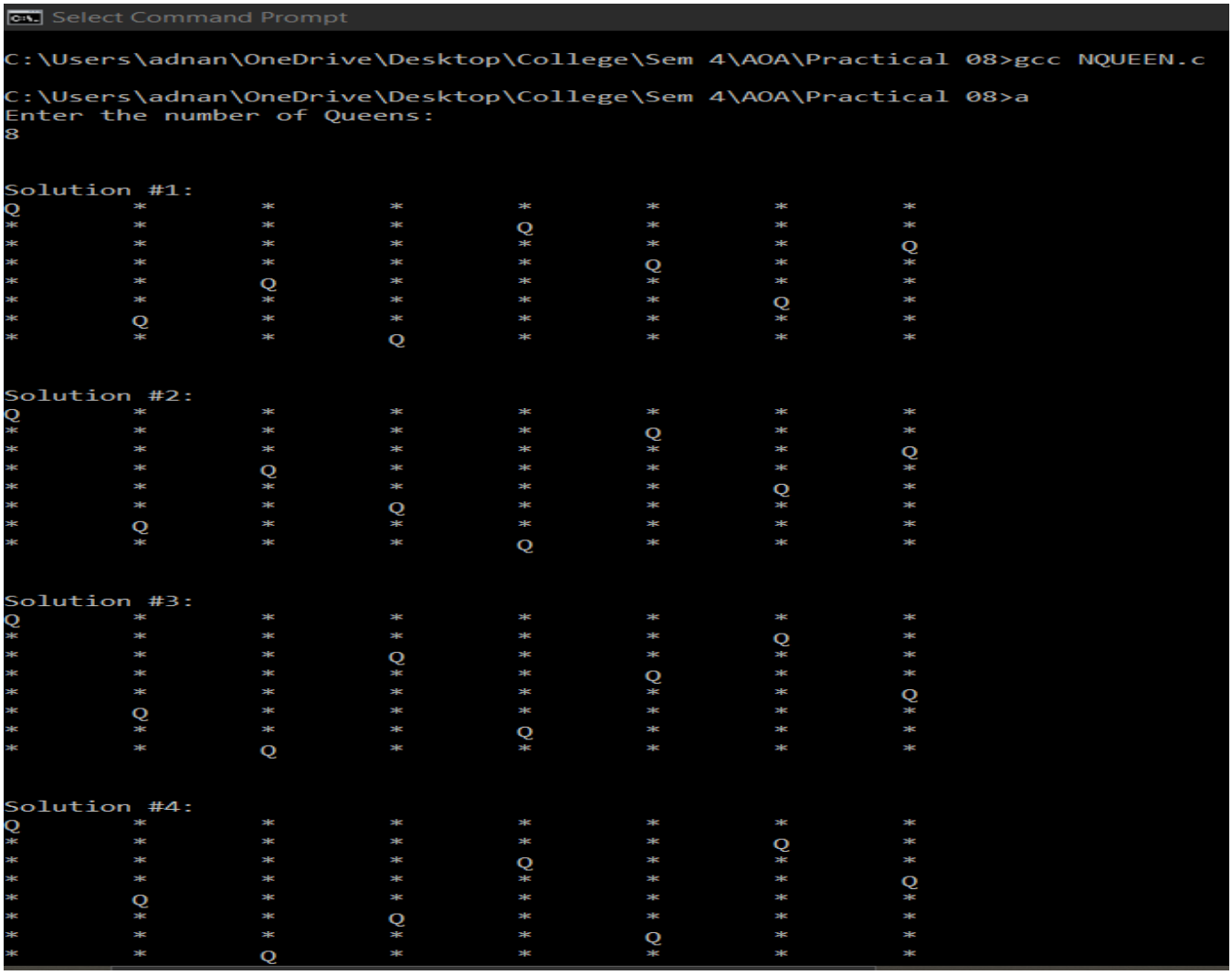
```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<stdlib.h>
int a[30],count=0;
int place(int pos) {
        int i;
        for (i=1;i<pos;i++) {
                if((a[i]==a[pos])||((abs(a[i]-a[pos])==abs(i-pos))))
                  return 0;
        }
        return 1;
}
void print_sol(int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

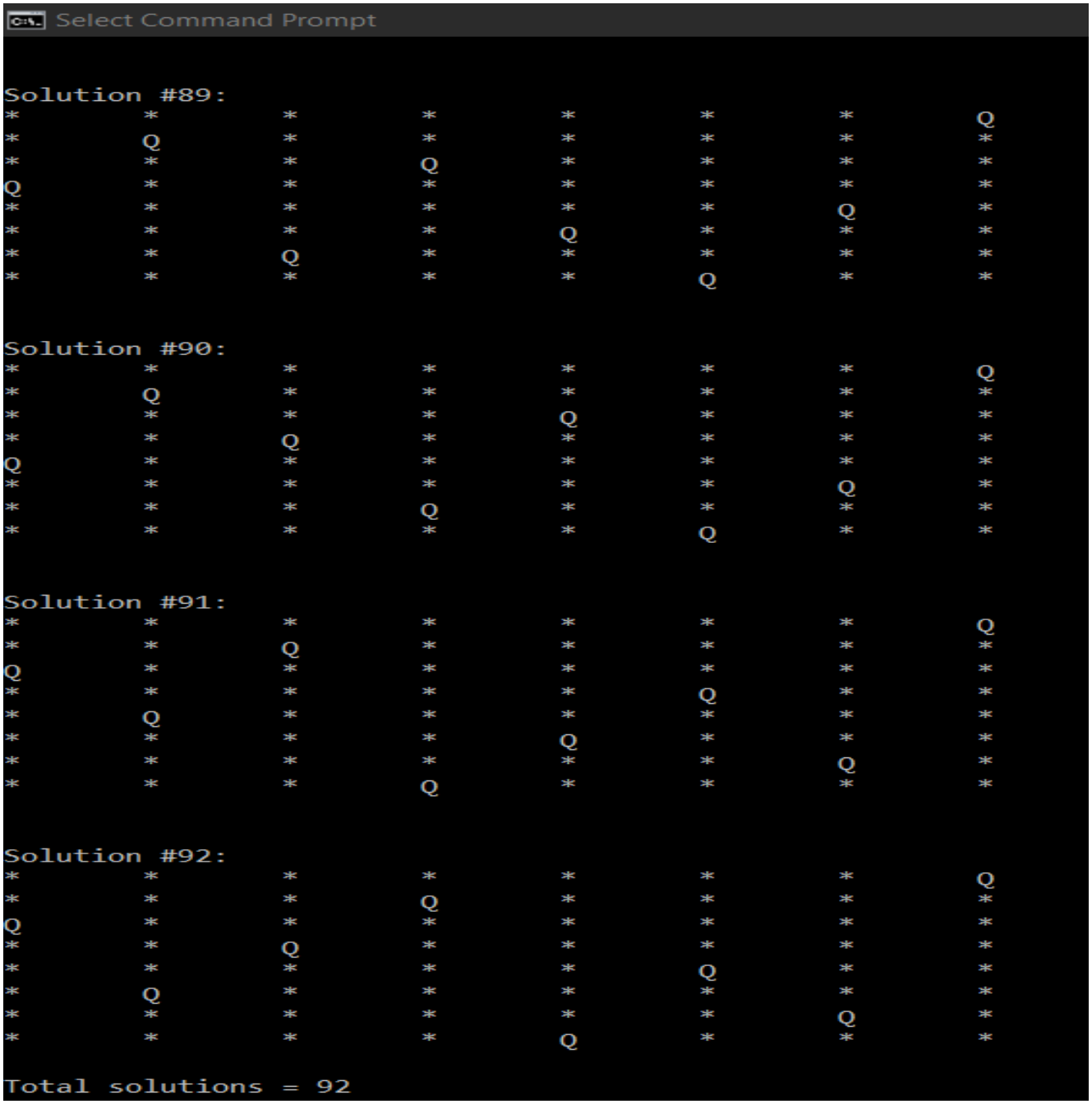
```
                        if(k==n)
                           print_sol(n); else {
                                 k++;
                                 a[k]=0;
                        }
                } else
                  k--;
        }
}
void main() {
        int i,n;

        printf("Enter the number of Queens: \n");
        scanf("%d",&n);
        queen(n);
        printf("\nTotal solutions = %d",count);

}
```

OUTPUT:





CONCLUSION: In backtracking technique, the eight-queen's problem is to place the eight queens checks against any other queen. The backtracking is applied till optimal solution is reached.