DAA Experiment-7

(Batch-A/A1)

Name	Ansari Mohammed Shanouf Valijan
UID Number	2021300004
Class	SY B.Tech Computer Engineering(Div-A)
Experiment Number	7
Date of Performance	13-04-23
Date of Submission	16-04-23

Aim:

To solve the N-Queen Problem using Backtracking strategy.

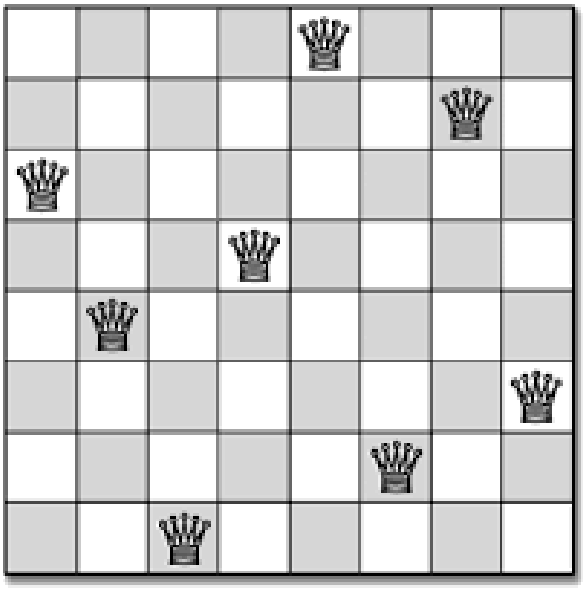
Theory:

The N-Queen problem is a classic computer science problem that involves placing N chess queens on an N x N chessboard in such a way that no two queens attack each other. This means that no two queens can share the same row, column, or diagonal on the board.

The problem can be approached using backtracking, which is a general algorithmic technique for finding all(or some) solutions to a problem that incrementally builds candidates to the solutions, and abandons a candidate ("backtracks") as soon as it determines that the candidate cannot possibly be completed to a valid solution.

The basic idea is to place queens on the board one at a time, starting from the topmost row. For each row, try placing a queen in each column until a safe placement is found. A placement is safe if and only if no two queens are attacking each other. If a safe placement is found, move on to the next row and repeat the process. If no safe placement is found, backtrack to the previous row and try the next column. If all columns have been tried in the first row, the problem is unsolvable.

Given below is one of the 92 solution instances possible for 8-Queen problem. The image can be examined to confirm that no two queens attack each other.



Using backtracking strategy to solve this problem would be extremely helpful as it will be possible to know all the feasible solutions rather than just one.

Algorithm:

[A] For solving the N-Queen problem-

I. Start.

II. Return from the function if the last queen is successfully placed on the ultimate row or required number of solutions have been obtained.

III. Perform the following steps for each row starting from the first one.

IV. Move the queen till such a position is found where it is not under attack.

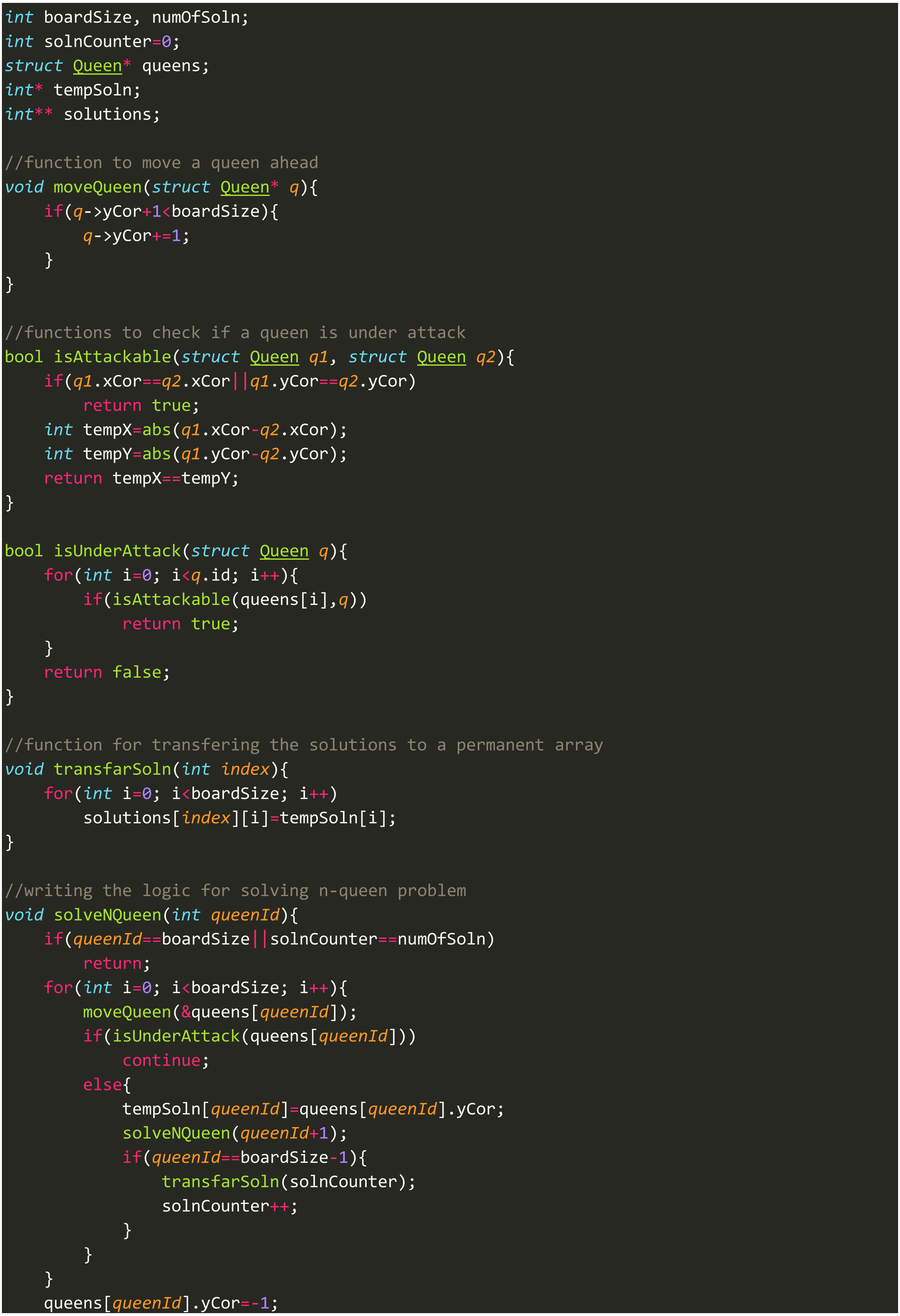
V. If no such position is found, backtrack to the previous row and repeat.

VI. If found, fix the queen's position and solve the N-Queen problem for the queen present in the next row.

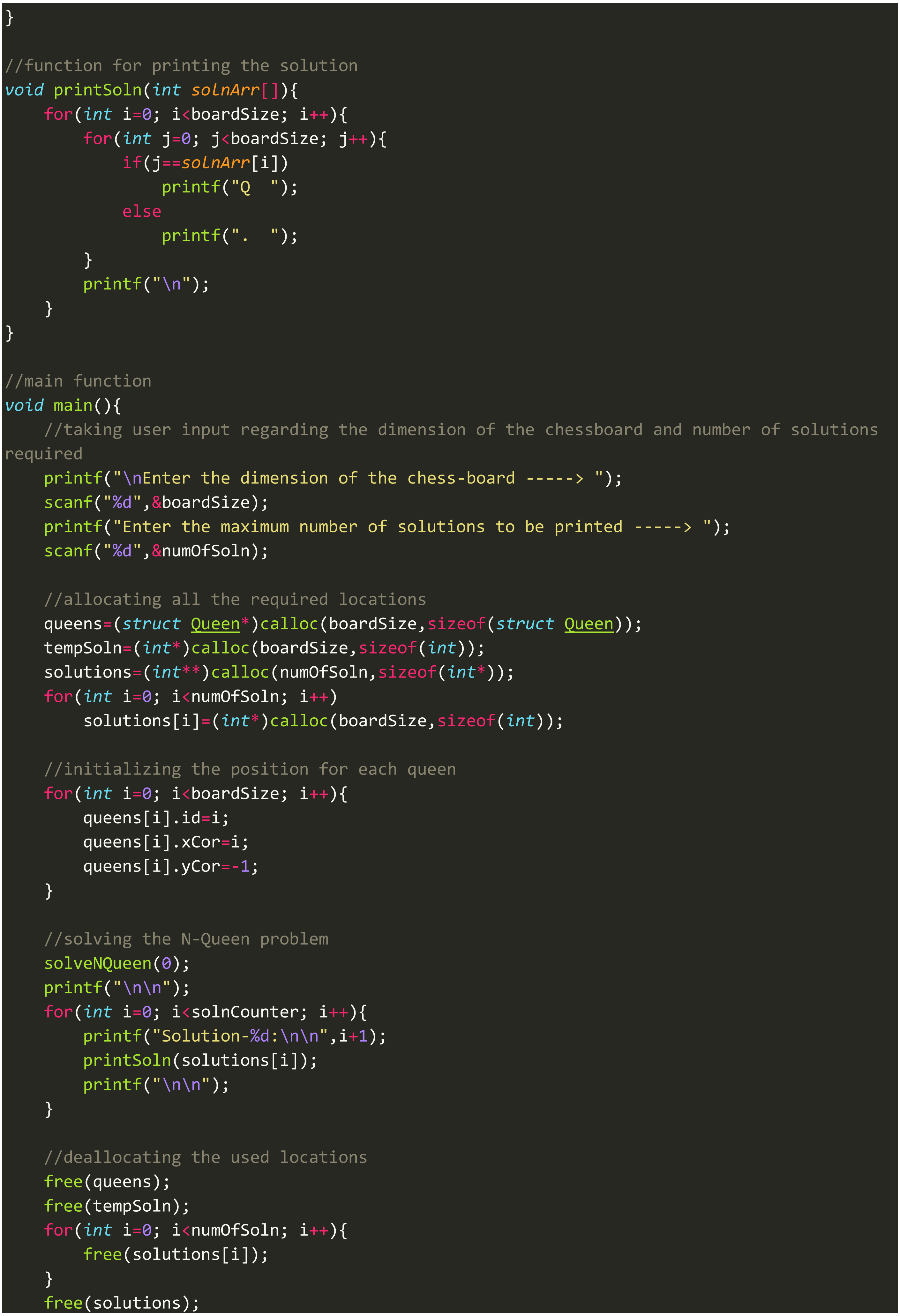
VII. End.

Program:

```c
//header files
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<stdbool.h>

//structure for storing queen info
struct Queen{
    int id;
    int xCor, yCor;
};

//global variables
```



```
int boardSize, numOfSoln;
int solnCounter=0;
struct Queen* queens;
int* tempSoln;
int** solutions;

//function to move a queen ahead
void moveQueen(struct Queen* q){
    if(q->yCor+1<boardSize){
        q->yCor+=1;
    }
}

//functions to check if a queen is under attack
bool isAttackable(struct Queen q1, struct Queen q2){
    if(q1.xCor==q2.xCor||q1.yCor==q2.yCor)
        return true;
    int tempX=abs(q1.xCor-q2.xCor);
    int tempY=abs(q1.yCor-q2.yCor);
    return tempX==tempY;
}

bool isUnderAttack(struct Queen q){
    for(int i=0; i<q.id; i++){
        if(isAttackable(queens[i],q))
            return true;
    }
    return false;
}

//function for transfering the solutions to a permanent array
void transfarSoln(int index){
    for(int i=0; i<boardSize; i++)
        solutions[index][i]=tempSoln[i];
}

//writing the logic for solving n-queen problem
void solveNQueen(int queenId){
    if(queenId==boardSize||solnCounter==numOfSoln)
        return;
    for(int i=0; i<boardSize; i++){
        moveQueen(&queens[queenId]);
        if(isUnderAttack(queens[queenId]))
            continue;
        else{
            tempSoln[queenId]=queens[queenId].yCor;
            solveNQueen(queenId+1);
            if(queenId==boardSize-1){
                transfarSoln(solnCounter);
                solnCounter++;
            }
        }
    }
    queens[queenId].yCor=-1;
```

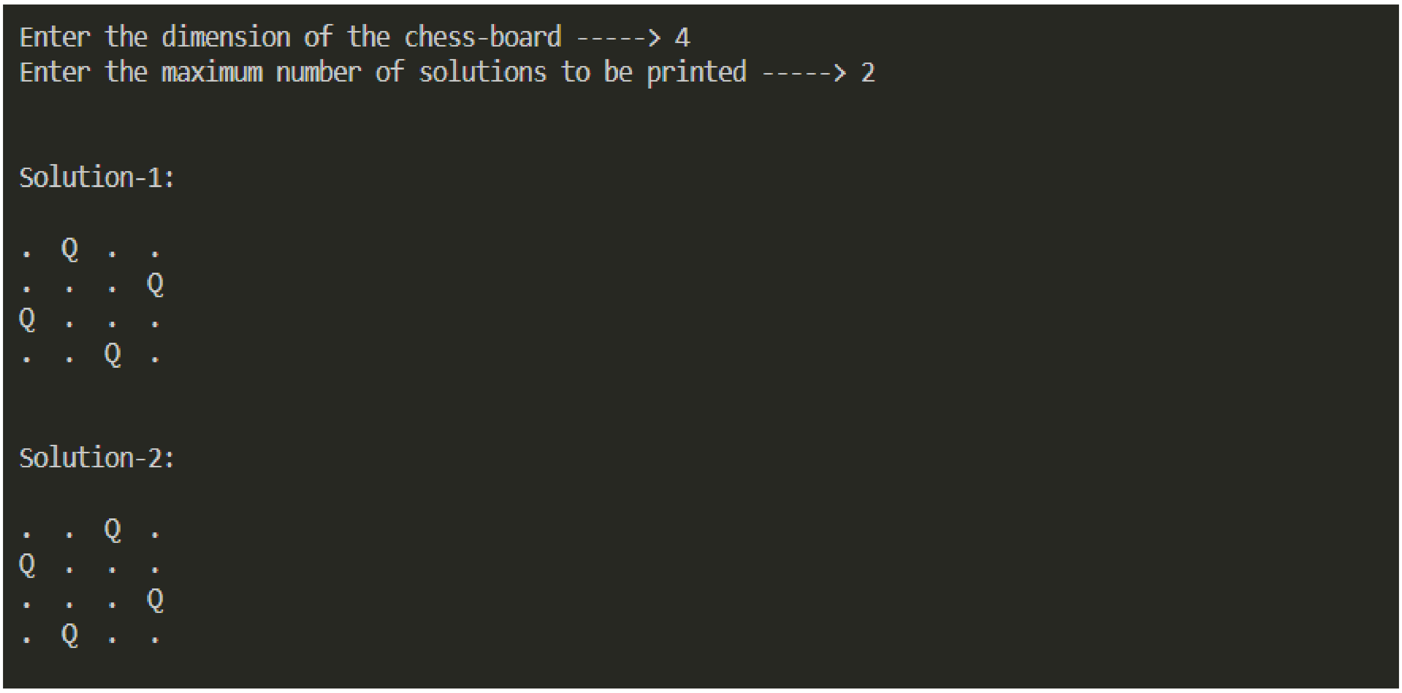


```
}

//function for printing the solution
void printSoln(int solnArr[]){
    for(int i=0; i<boardSize; i++){
        for(int j=0; j<boardSize; j++){
            if(j==solnArr[i])
                printf("Q  ");
            else
                printf(".  ");
        }
        printf("\n");
    }
}

//main function
void main(){
    //taking user input regarding the dimension of the chessboard and number of solutions
required
    printf("\nEnter the dimension of the chess-board -----> ");
    scanf("%d",&boardSize);
    printf("Enter the maximum number of solutions to be printed -----> ");
    scanf("%d",&numOfSoln);

    //allocating all the required locations
    queens=(struct Queen*)calloc(boardSize,sizeof(struct Queen));
    tempSoln=(int*)calloc(boardSize,sizeof(int));
    solutions=(int**)calloc(numOfSoln,sizeof(int*));
    for(int i=0; i<numOfSoln; i++)
        solutions[i]=(int*)calloc(boardSize,sizeof(int));

    //initializing the position for each queen
    for(int i=0; i<boardSize; i++){
        queens[i].id=i;
        queens[i].xCor=i;
        queens[i].yCor=-1;
    }

    //solving the N-Queen problem
    solveNQueen(0);
    printf("\n\n");
    for(int i=0; i<solnCounter; i++){
        printf("Solution-%d:\n\n",i+1);
        printSoln(solutions[i]);
        printf("\n\n");
    }

    //deallocating the used locations
    free(queens);
    free(tempSoln);
    for(int i=0; i<numOfSoln; i++){
        free(solutions[i]);
    }
    free(solutions);
```

```
}
```

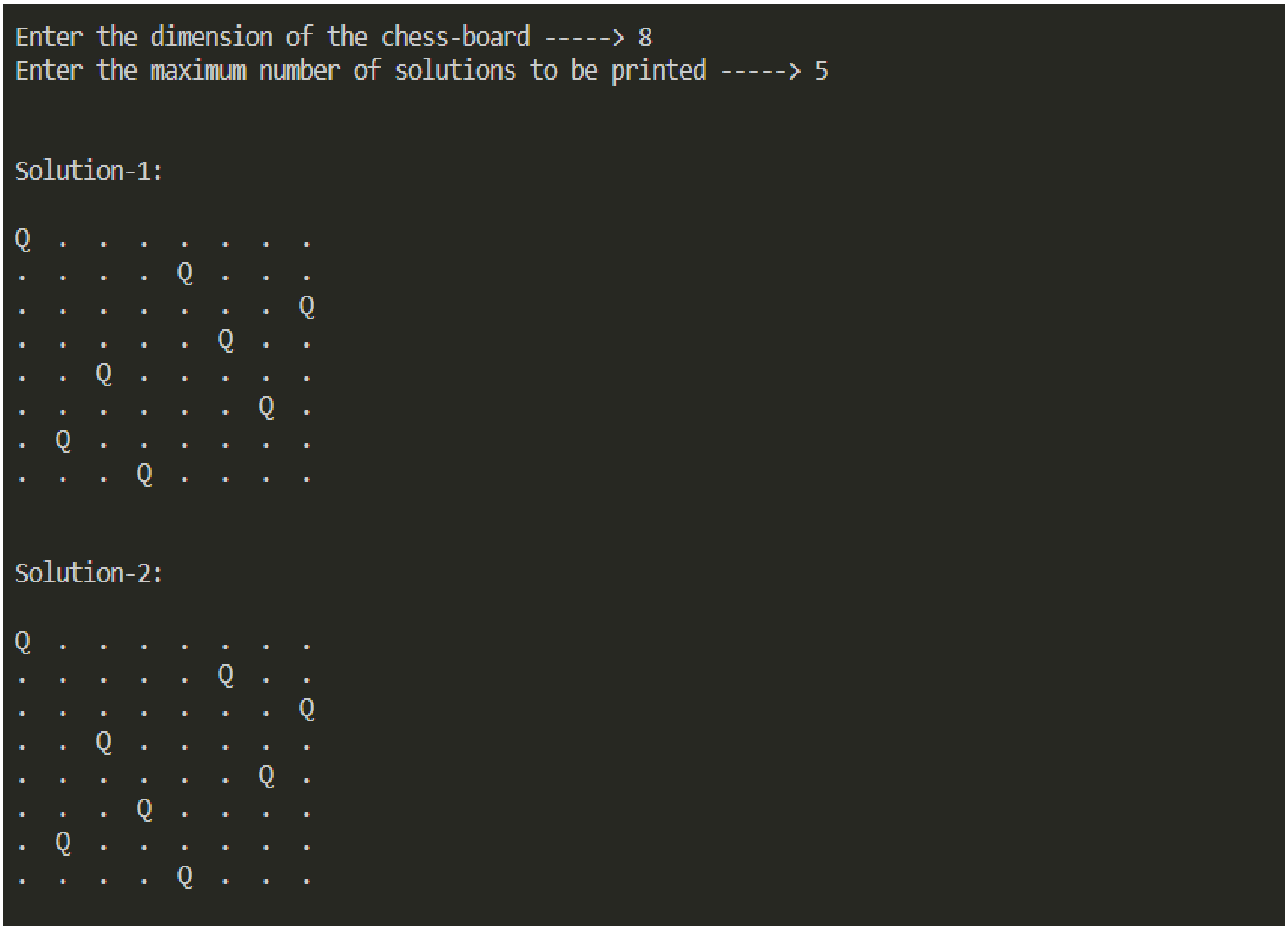
Implementation:

4-Queen-problem-

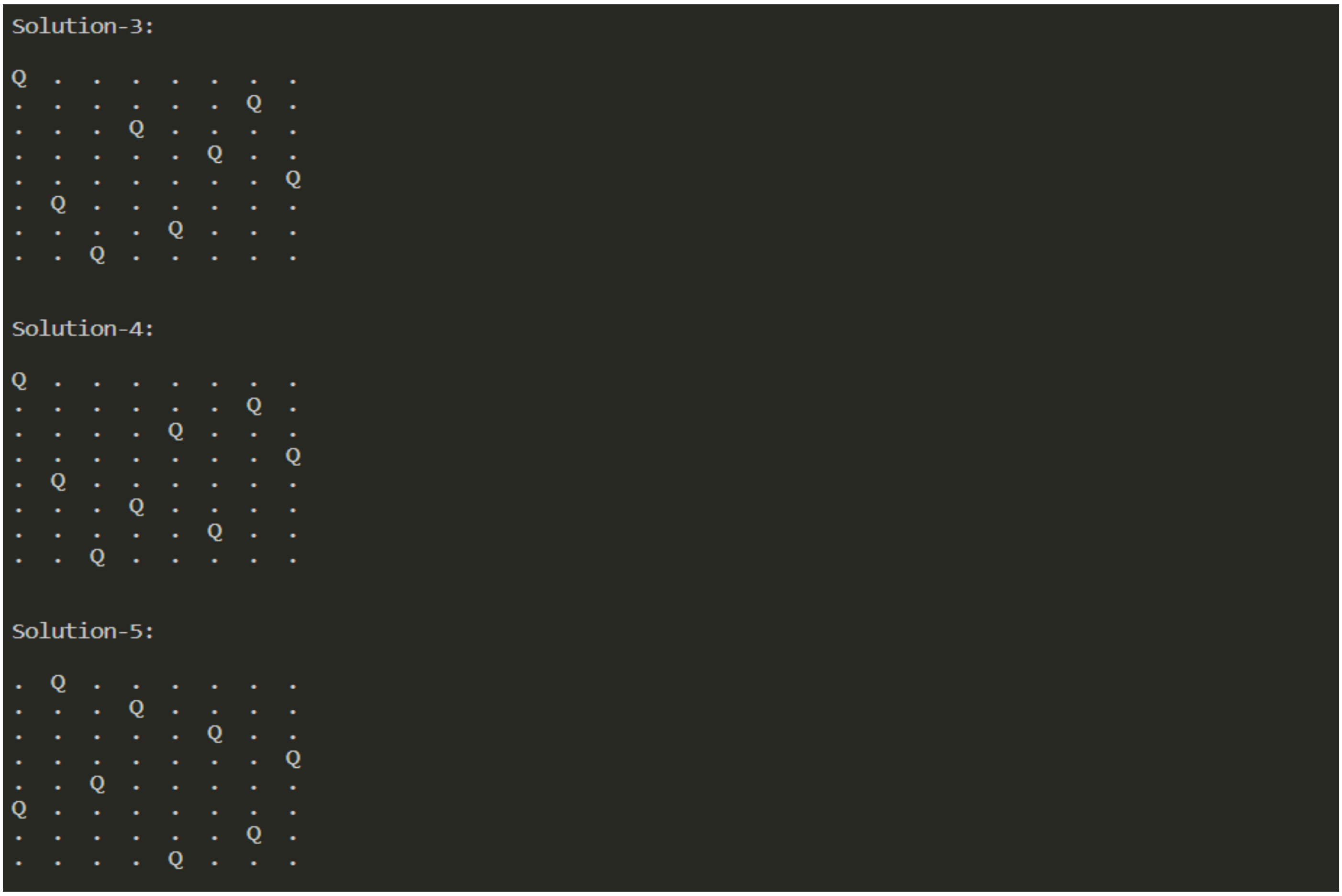


```
Enter the dimension of the chess-board -----> 4
Enter the maximum number of solutions to be printed -----> 2


Solution-1:

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


Solution-2:

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

8-Queen-problem-

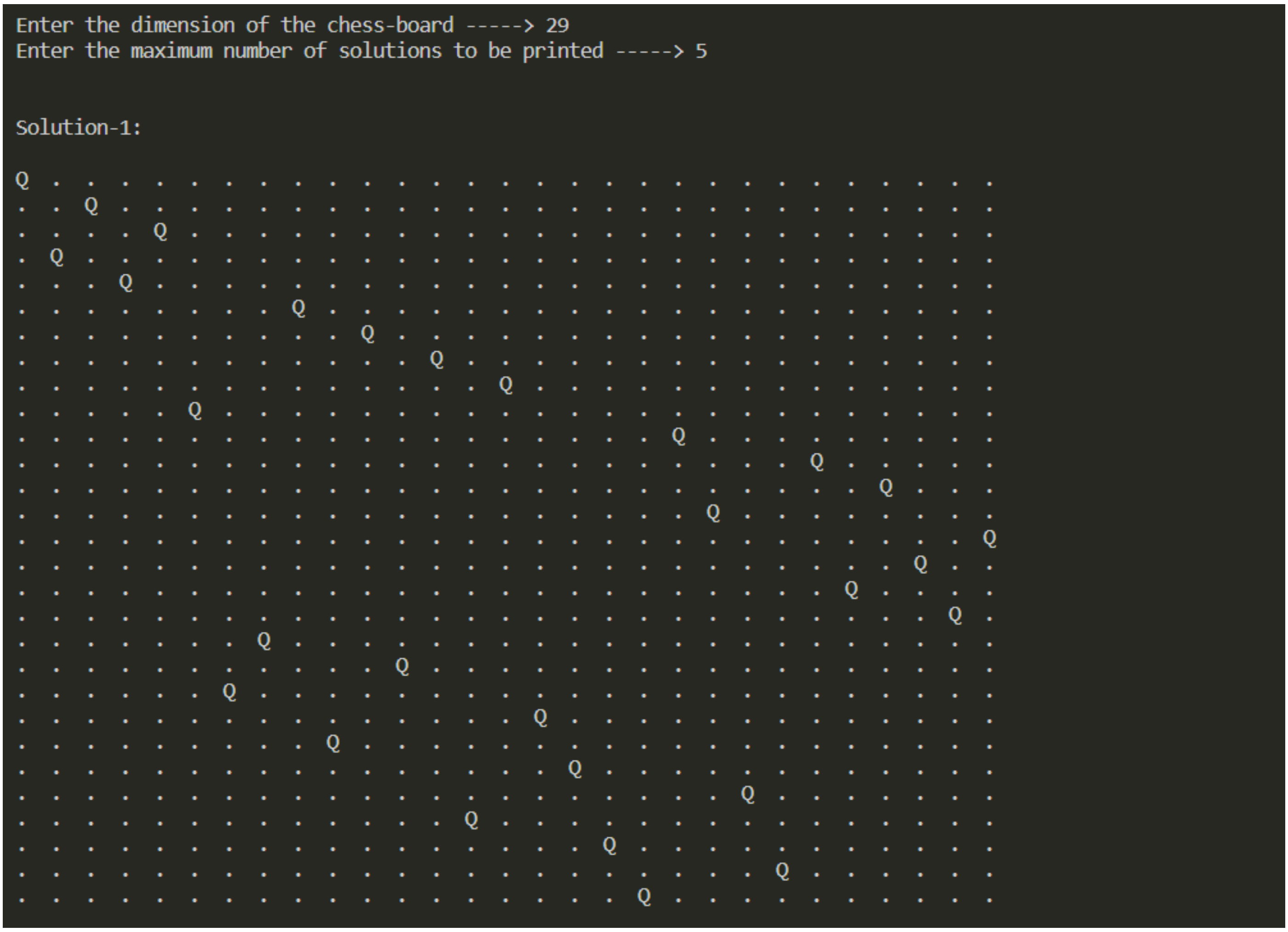


```
Enter the dimension of the chess-board -----> 8
Enter the maximum number of solutions to be printed -----> 5


Solution-1:

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


Solution-2:

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

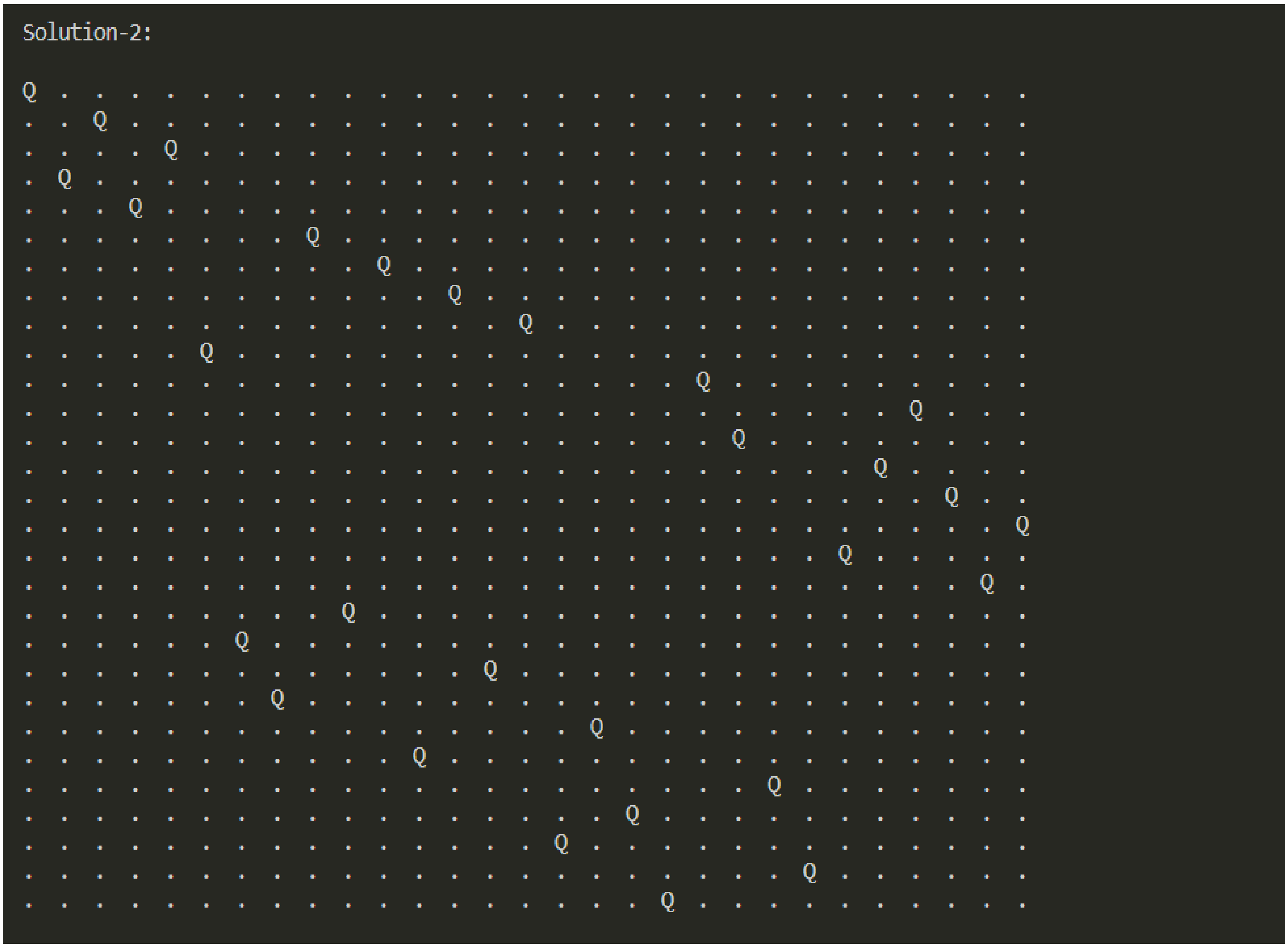


```
Solution-3:

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 . . .
```

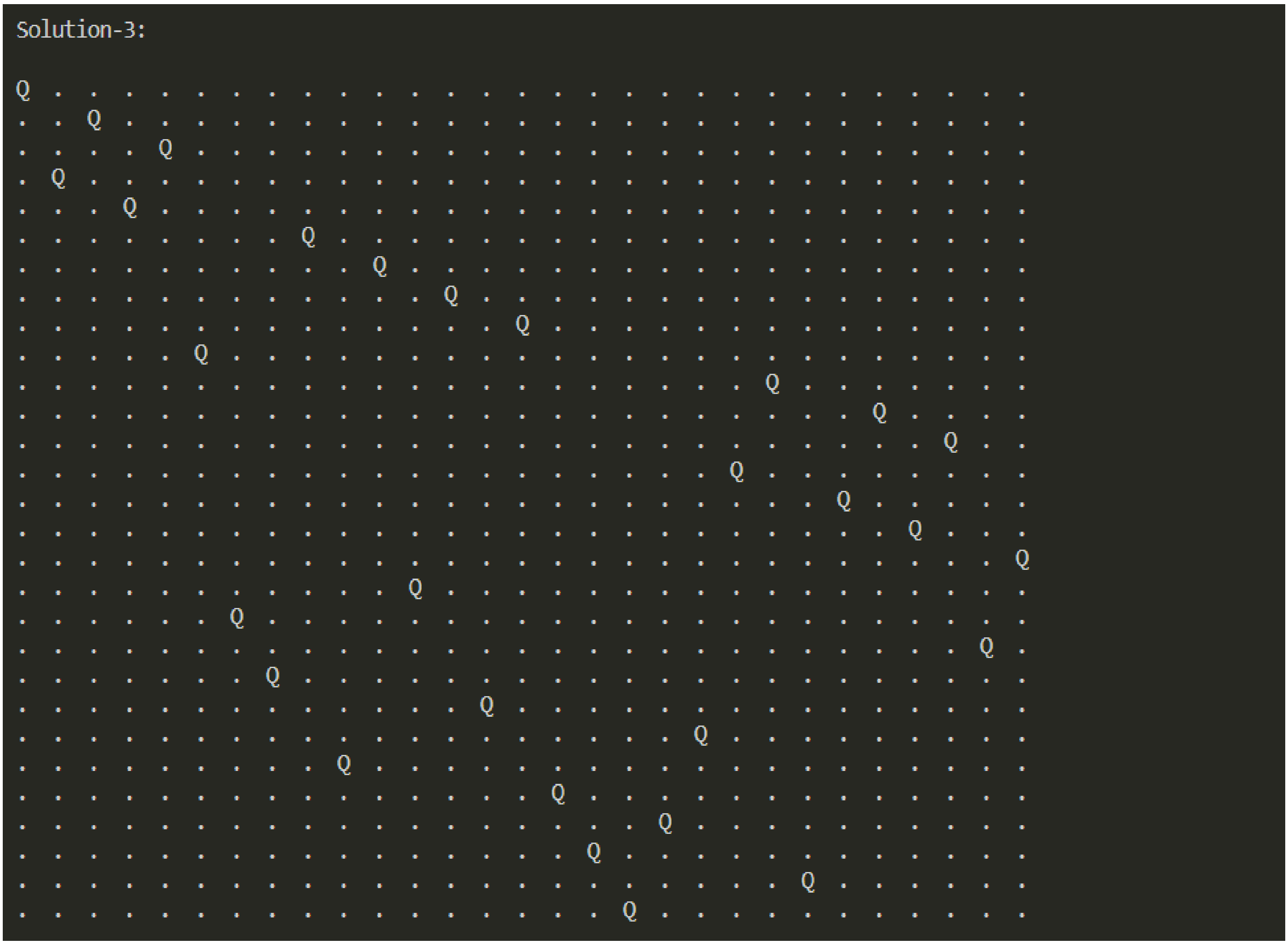
29-Queen-problem-



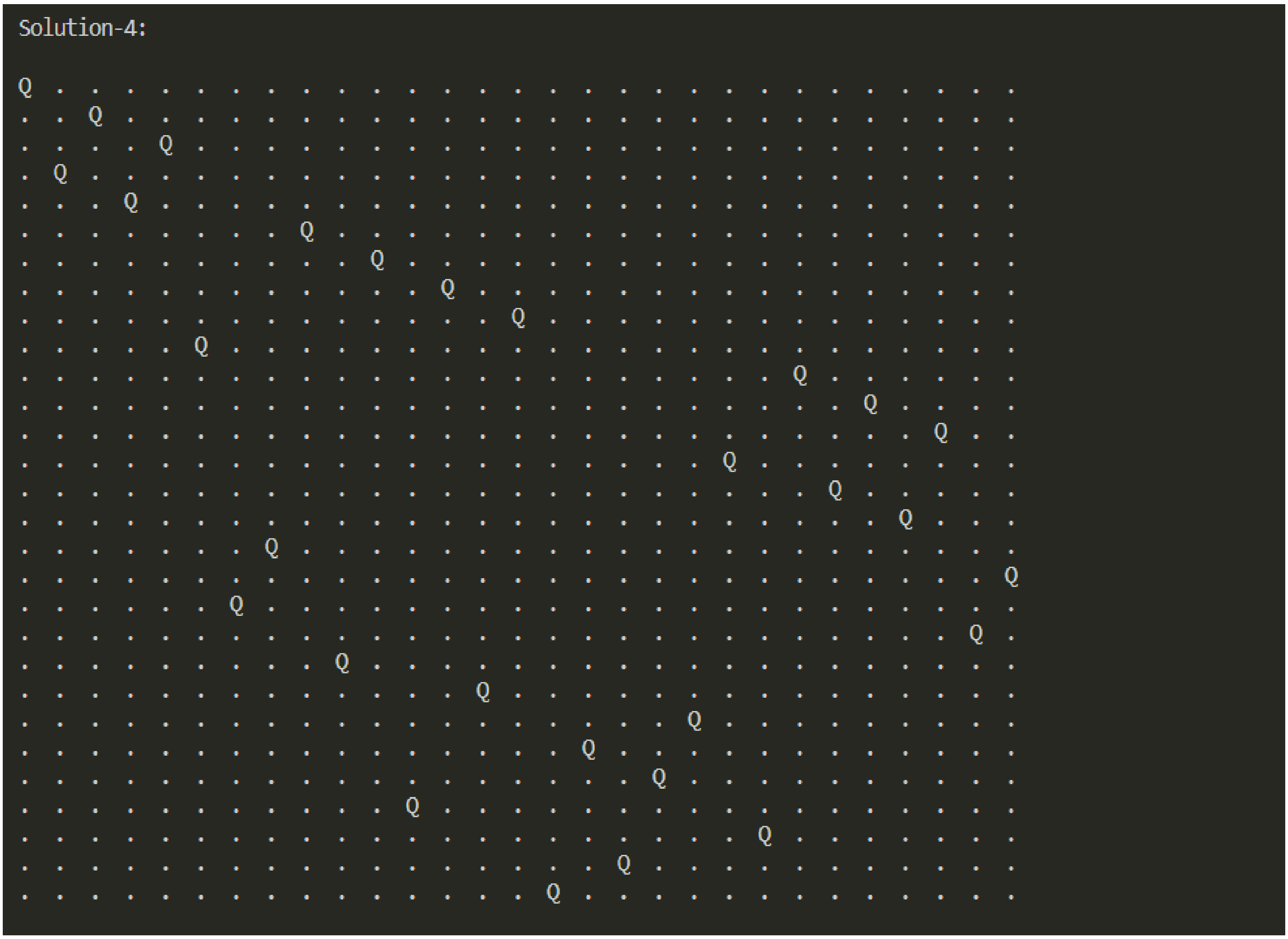
```
Enter the dimension of the chess-board -----> 29
Enter the maximum number of solutions to be printed -----> 5


Solution-1:

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



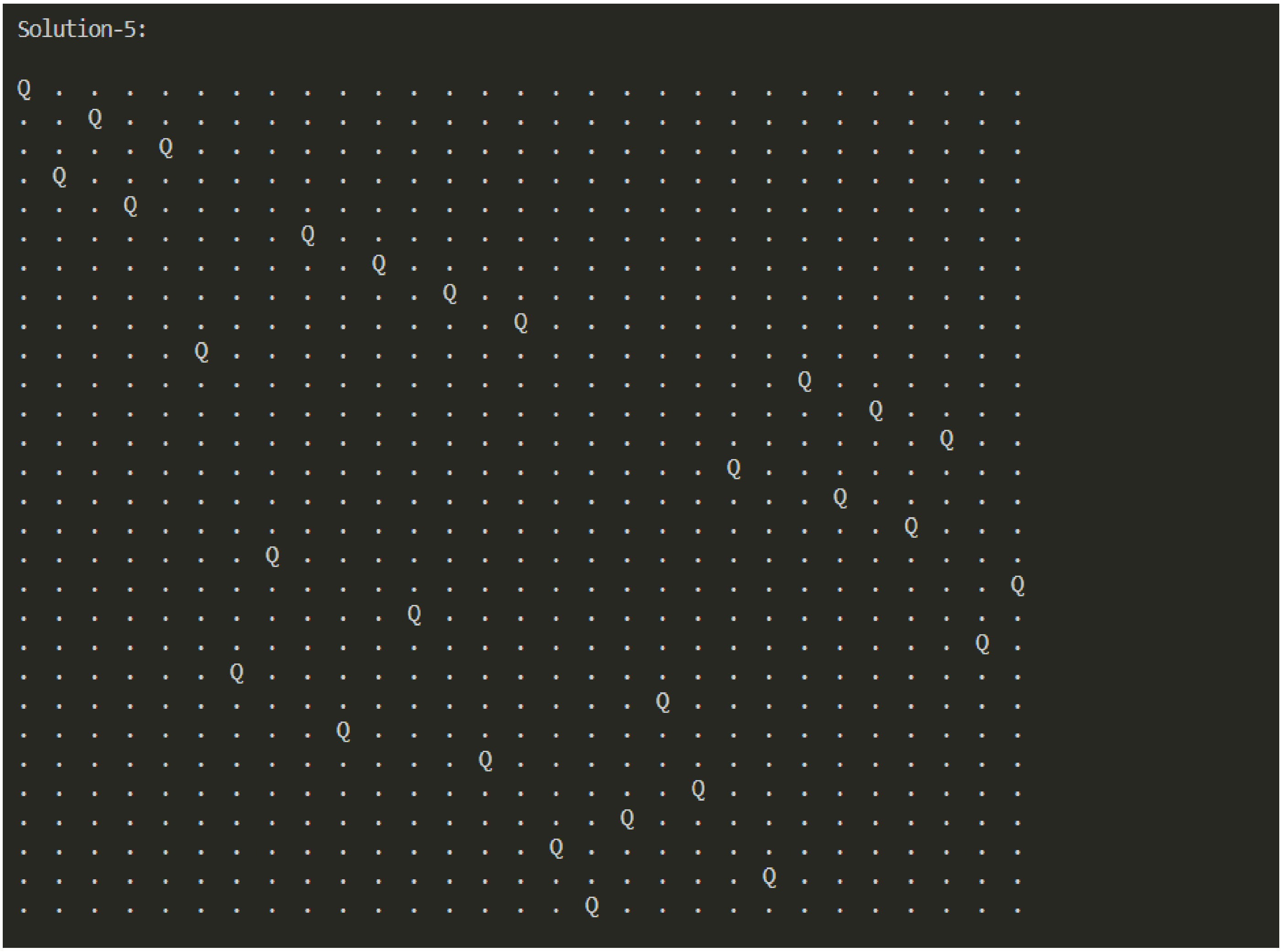
```
Solution-2:

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



```
Solution-3:

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

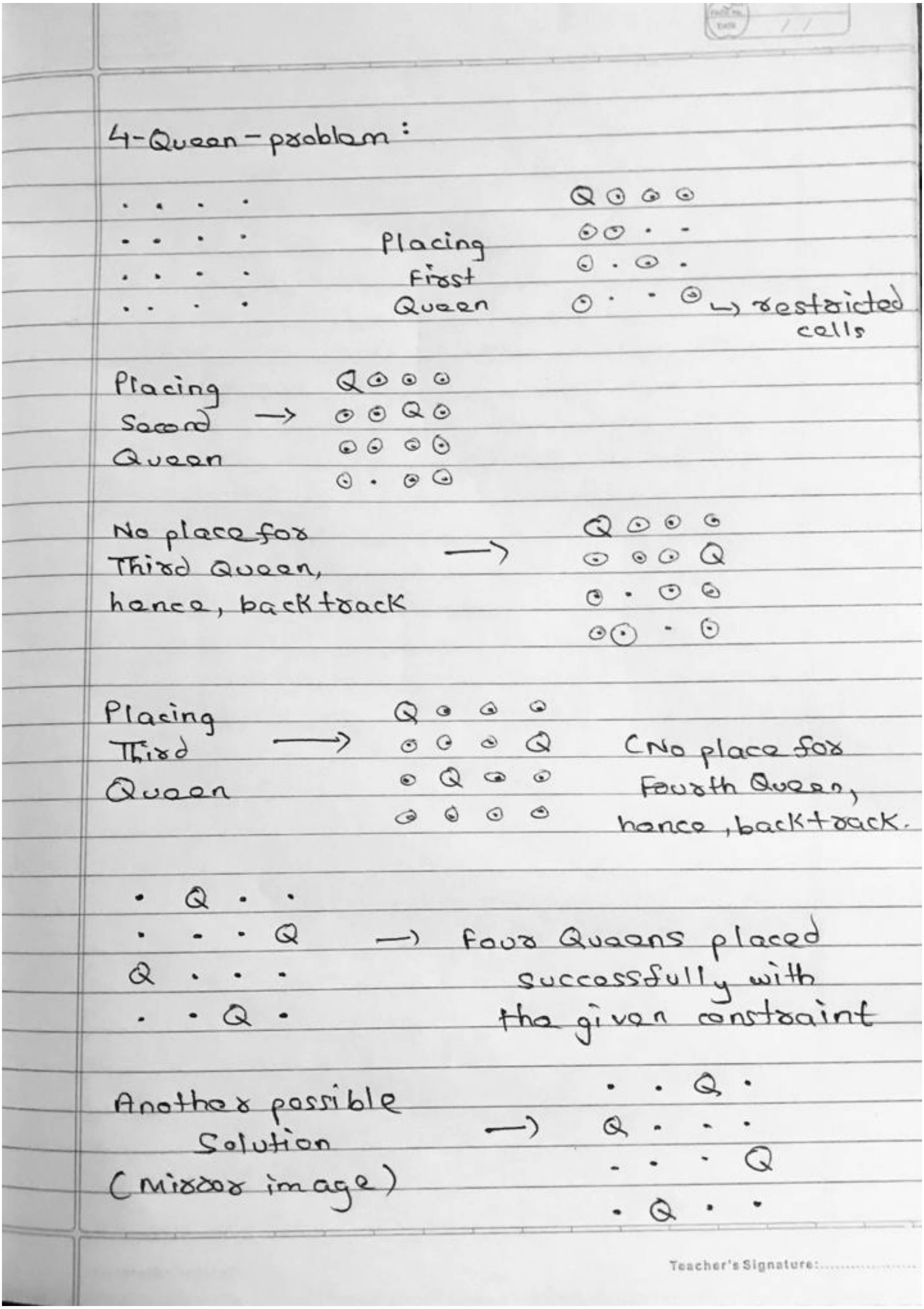


```
Solution-4:

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



```
Solution-5:

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

Following example of 4-Queen problem was solved to understand backtracking-



4-Queen-problem:

```
· · · ·                    Q ⊙ ⊙ ⊙
· · · ·       Placing      ⊙ ⊙ · ·
· · · ·       First        ⊙ · ⊙ ·
· · · ·       Queen        ⊙ · · ⊙  → restricted
                                        cells
```

```
Placing        Q ⊙ ⊙ ⊙
Second   →     ⊙ ⊙ Q ⊙
Queen          ⊙ ⊙ ⊙ ⊙
               ⊙ · ⊙ ⊙
```

```
No place for                  Q ⊙ ⊙ ⊙
Third Queen,          →       ⊙ ⊙ ⊙ Q
hence, backtrack              ⊙ · ⊙ ⊙
                              ⊙ ⊙ · ⊙
```

```
Placing          Q ⊙ ⊙ ⊙
Third     →      ⊙ ⊙ ⊙ Q       (No place for
Queen            ⊙ Q ⊙ ⊙        Fourth Queen,
                 ⊙ ⊙ ⊙ ⊙       hence, backtrack.
```

```
· Q · ·
· · · Q    →   Four Queens placed
Q · · ·          successfully with
· · Q ·        the given constraint
```

```
                               · · Q ·
Another possible               Q · · ·
   Solution           →        · · · Q
(Mirror image)                 · Q · ·
```

Teacher's Signature:.....................

Inference:

While coding the program, I realized that recursion depth for solving the N-Queen problem would be N as in each function call, we deal with one row for finding the suitable position. Once found, we fix it and move to the next row through recursion to repeat the same. While executing the program, I got to know that it displays the results for 29 queens in no time. However, for further counts, it takes some time. This might depend on the processing power of the device as well as the available memory.

Conclusion:

By performing this experiment, I was able to understand backtracking strategy. I was able to realize that one can use backtracking if all the feasible solutions to a problem is required and I was also able to solve N-Queen problem using the said strategy.