Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

JHANVI PAREKH

60009210033

CSE(Data Science)

Experiment 9 (Backtracking)

Aim: Implementation of 8 queen problem.

Theory:

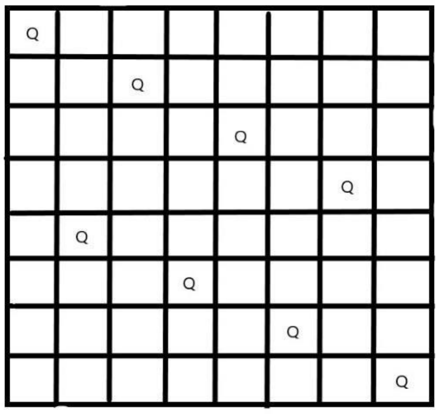
You are given an 8x8 chessboard, find a way to place 8 queens such that no queen can attack any other queen on the chessboard. A queen can only be attacked if it lies on the same row, or same column, or the same diagonal of any other queen. Print all the possible configurations.

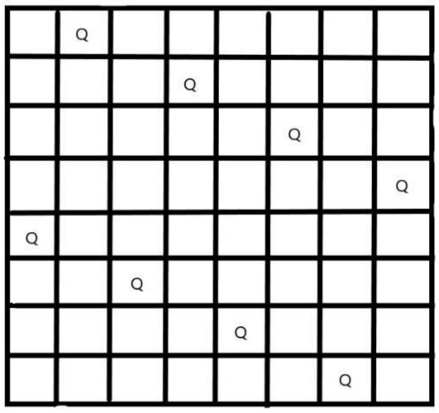
To solve this problem, we will make use of the **Backtracking algorithm**. The backtracking algorithm, in general checks all possible configurations and test whether the required result is obtained or not. For thr given problem, we will explore all possible positions the queens can be relatively placed at. The solution will be correct when the number of placed queens = 8.

Input Format - the number 8, which does not need to be read, but we will take an input number for the sake of generalization of the algorithm to an NxN chessboard.

Output Format - all matrices that constitute the possible solutions will contain the numbers 0(for empty cell) and 1(for a cell where queen is placed). Hence, the output is a set of binary matrices.

Example:





Algorithm:

1. Begin from the leftmost column

2. If all the queens are placed, return true/ print the matrix
3. Check for all rows in the current column
    a) if queen placed safely, mark row and column; and recursively check if we approach in the current configuration, do we obtain a solution or not

    b) if placing yields a solution, return true
    c) if placing does not yield a solution, unmark and try other rows
4. If all rows tried and solution not obtained, return false and backtrack

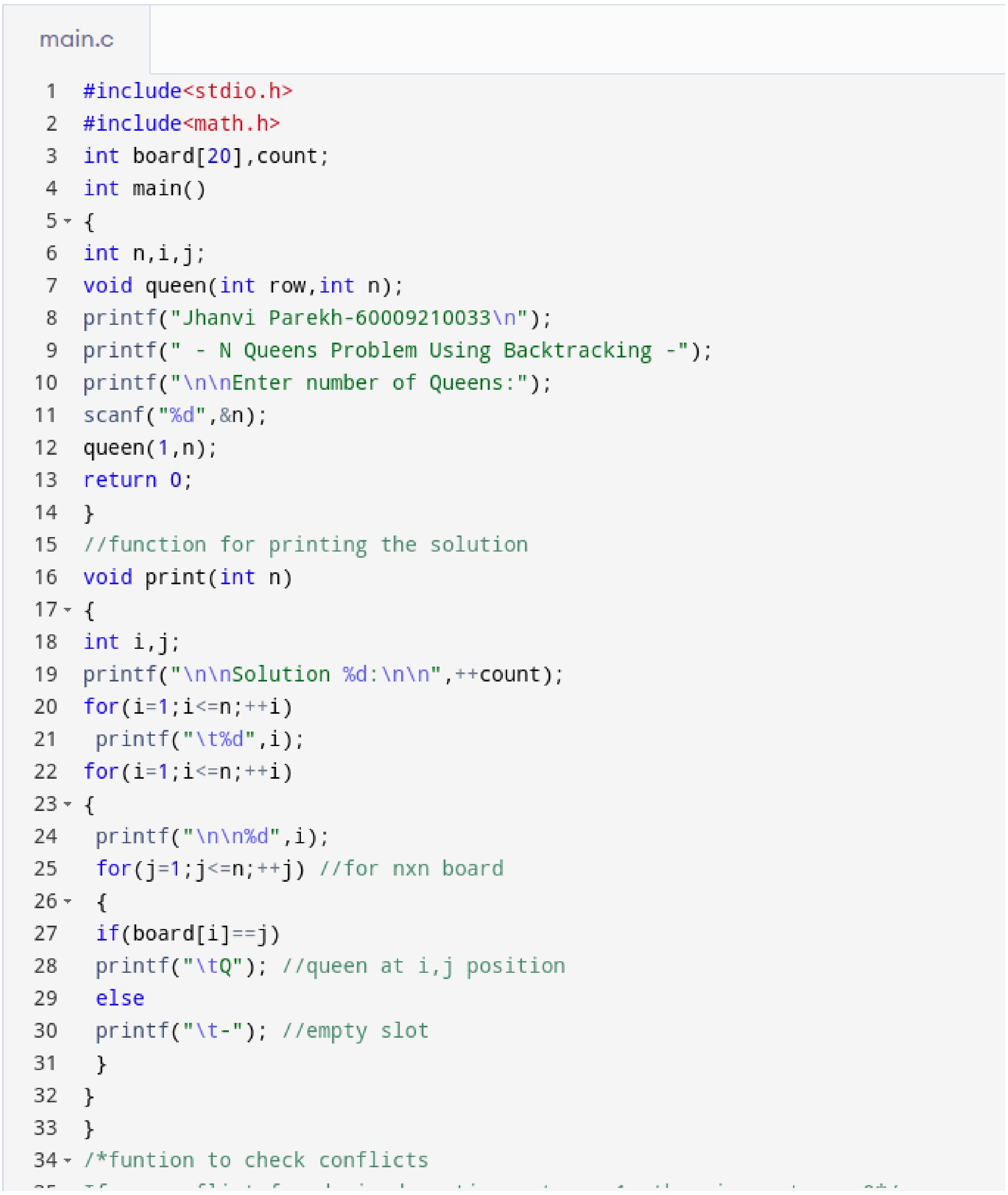
Pseudocode:

```
Algorithm N_QUEEN (k, n)
// Description : To find the solution of n x n queen problem using backtracking //
Input :
n: Number of queen k: Number of the queen being processed
currently, initially set to 1.

// Output : n x 1 Solution tuple


for i ← 1 to n do   if
PLACE(k , i) then
x[k] ← i     if k == n
then
    print X[1…n]
else
    N_QUEEN(k + 1, n)
end   end end

Function PLACE(k, i)
// k is the number of queen being processed
// i is the number of columns


for j ← 1 to k – 1 do
  if x[j] == i OR ((abs(x[j]) - i) == abs(j - k)) then
return false   end end
return true
```

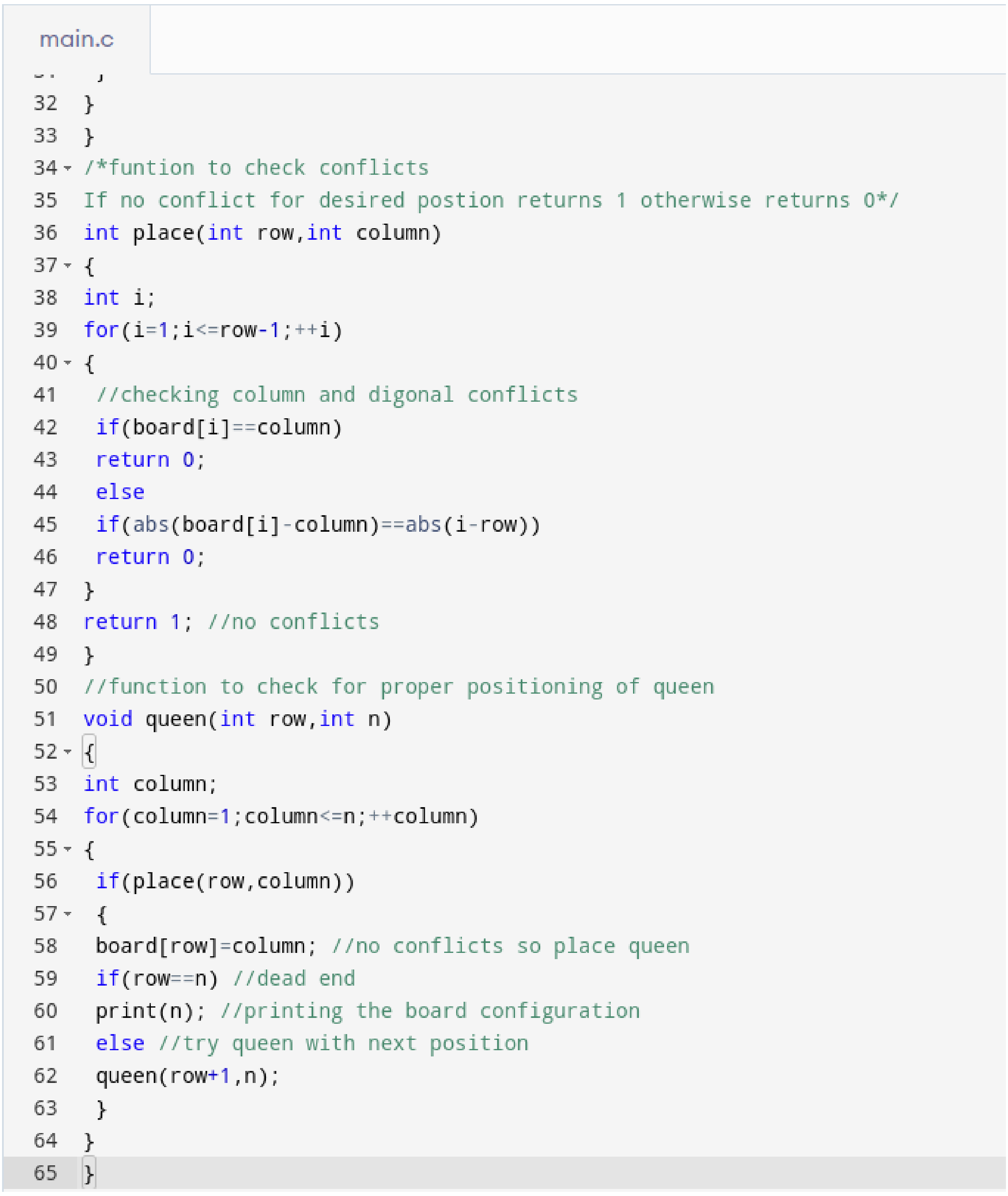
Time Complexity: O(N!)

Lab Assignment to Complete:

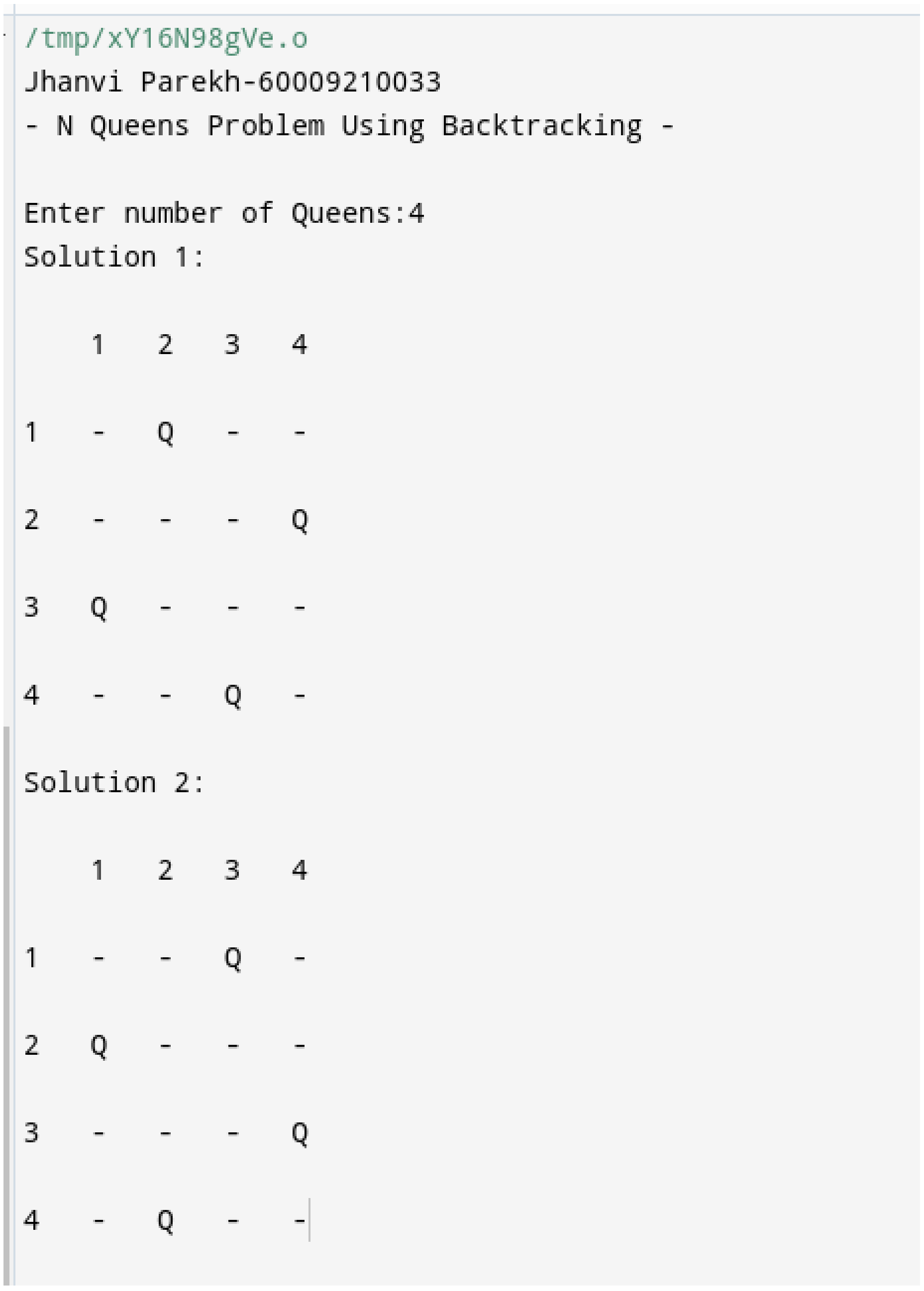
Using the above code and find a possible solution for N-queen problem, where N=4, 6, 8

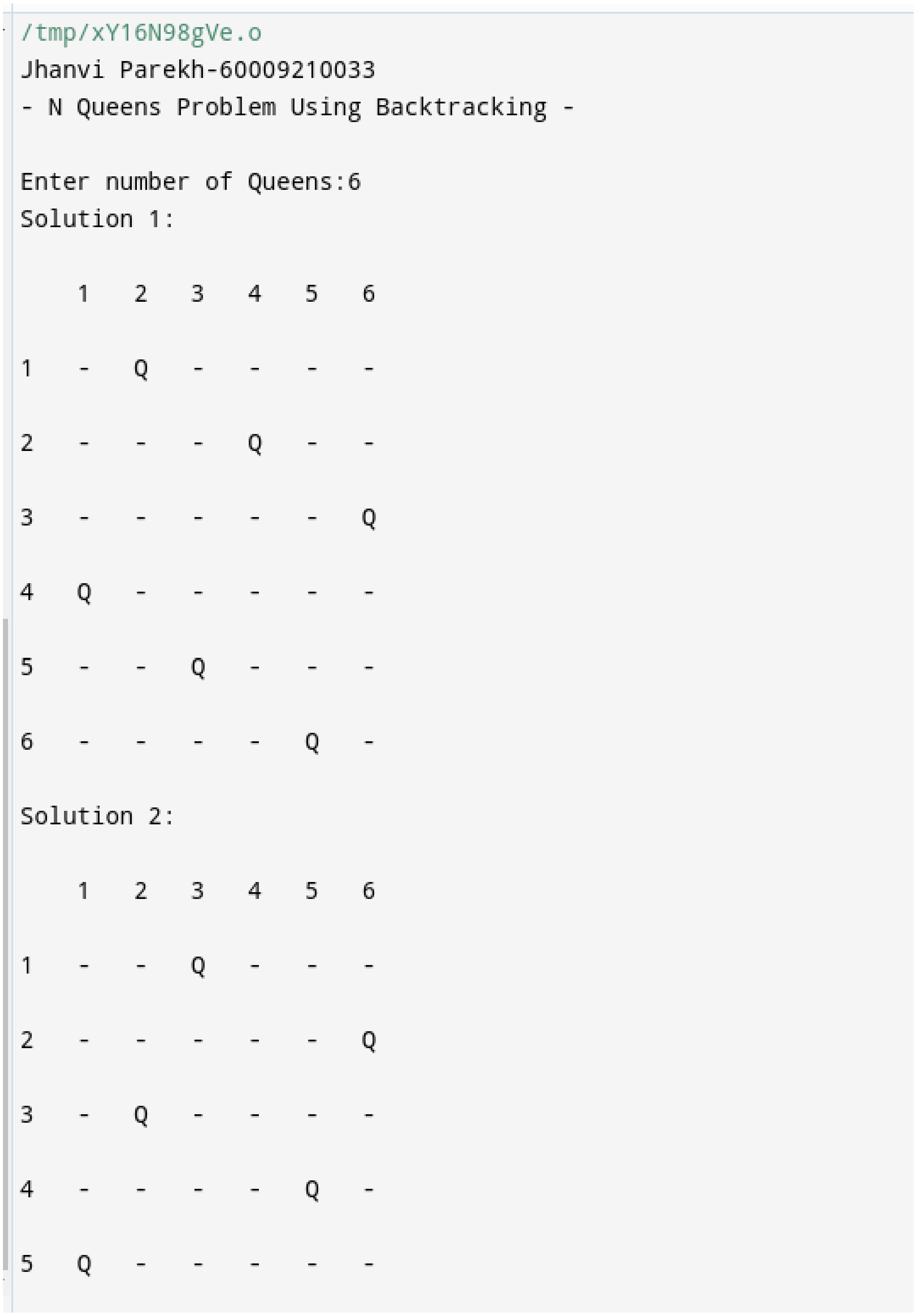


```c
#include<stdio.h>
#include<math.h>
int board[20],count;
int main()
{
int n,i,j;
void queen(int row,int n);
printf("Jhanvi Parekh-60009210033\n");
printf(" - N Queens Problem Using Backtracking -");
printf("\n\nEnter number of Queens:");
scanf("%d",&n);
queen(1,n);
return 0;
}
//function for printing the solution
void print(int n)
{
int i,j;
printf("\n\nSolution %d:\n\n",++count);
for(i=1;i<=n;++i)
 printf("\t%d",i);
for(i=1;i<=n;++i)
{
 printf("\n\n%d",i);
 for(j=1;j<=n;++j) //for nxn board
 {
 if(board[i]==j)
 printf("\tQ"); //queen at i,j position
 else
 printf("\t-"); //empty slot
 }
}
}
/*funtion to check conflicts
```



```c
 }
}
}
/*funtion to check conflicts
If no conflict for desired postion returns 1 otherwise returns 0*/
int place(int row,int column)
{
int i;
for(i=1;i<=row-1;++i)
{
 //checking column and digonal conflicts
 if(board[i]==column)
 return 0;
 else
 if(abs(board[i]-column)==abs(i-row))
 return 0;
}
return 1; //no conflicts
}
//function to check for proper positioning of queen
void queen(int row,int n)
{
int column;
for(column=1;column<=n;++column)
{
 if(place(row,column))
 {
 board[row]=column; //no conflicts so place queen
 if(row==n) //dead end
 print(n); //printing the board configuration
 else //try queen with next position
 queen(row+1,n);
 }
}
}
```


```
/tmp/xY16N98gVe.o
Jhanvi Parekh-60009210033
- N Queens Problem Using Backtracking -

Enter number of Queens:4
Solution 1:


    1   2   3   4


1   -   Q   -   -


2   -   -   -   Q


3   Q   -   -   -


4   -   -   Q   -


Solution 2:


    1   2   3   4


1   -   -   Q   -


2   Q   -   -   -


3   -   -   -   Q


4   -   Q   -   -
```



```
/tmp/xY16N98gVe.o
Jhanvi Parekh-60009210033
- N Queens Problem Using Backtracking -

Enter number of Queens:6
Solution 1:

    1   2   3   4   5   6

1   -   Q   -   -   -   -

2   -   -   -   Q   -   -

3   -   -   -   -   -   Q

4   Q   -   -   -   -   -

5   -   -   Q   -   -   -

6   -   -   -   -   Q   -

Solution 2:

    1   2   3   4   5   6

1   -   -   Q   -   -   -

2   -   -   -   -   -   Q

3   -   Q   -   -   -   -

4   -   -   -   -   Q   -

5   Q   -   -   -   -   -
```

```
5   Q

6   -   -   -   Q   -   -

Solution 3:

    1   2   3   4   5   6

1   -   -   -   Q   -   -

2   Q   -   -   -   -   -

3   -   -   -   -   Q   -

4   -   Q   -   -   -   -

5   -   -   -   -   -   Q

6   -   -   Q   -   -   -

Solution 4:

    1   2   3   4   5   6

1   -   -   -   -   Q   -

2   -   -   Q   -   -   -

3   Q   -   -   -   -   -

4   -   -   -   -   -   Q

5   -   -   -   Q   -   -

6   -   Q   -   -   -   -
```

Backtracking.

N-Queen Problem

N = 4

2 solutions

	1	2	3	4
1		Q		
2				Q
3	Q			
4			Q	

	1	2	3	4
1			Q	
2	Q			
3				Q
4		Q		

N = 6 ∴ 4 solutions

	1	2	3	4	5	6
1		Q				
2				Q		
3						Q
4	Q					
5			Q			
6					Q	

	1	2	3	4	5	6
1			Q			
2		Q				Q
3		Q				
4					Q	
5	Q					
6				Q		

	1	2	3	4	5	6
1				Q		
2	Q					
3					Q	
4		Q				
5						Q
6			Q			

	1	2	3	4	5	6
1					Q	
2			Q			
3	Q					
4						Q
5				Q		
6		Q				

N=8 for this we will have 92 solutions

for ex:

	1	2	3	4	5	6	7	8
1	Q							
2					Q			
3								Q
4						Q		
5			Q	~~Q~~				
6							Q	
7		Q						
8				Q				