Experiment No: 9

Name: Shawn Louis

Roll No: 31

Batch: B

Problem Statement:

To implement N Queen Problem

Using Backtracking Strategy

Objective:

- To be able to implement a problem using backtracking strategy

Expected Outcome:

- **Ability to explain the problem statement**
- **Ability to build a puzzle using the specified strategy**
- **Ability to understand the use of recursion in backtracking.**

Theory:

This problem is to find an arrangement of N queens on a chess board, such that no queen can attack any other queens on the board.

The chess queens can attack in any direction as horizontal, vertical, horizontal and diagonal way.

A binary matrix is used to display the positions of N Queens, where no queens can attack other queens.

Algorithm:

```
Queen(row, n):
Begin
   if all columns are filled, then
      return true
   for each row of the board, do
      if isValid(board, i, col), then
         set queen at place (i, col) in the board
         if solveNQueen(board, col+1) = true, then
            return true
         otherwise remove queen from place (i, col) from
board.
   done
   return false
End
```

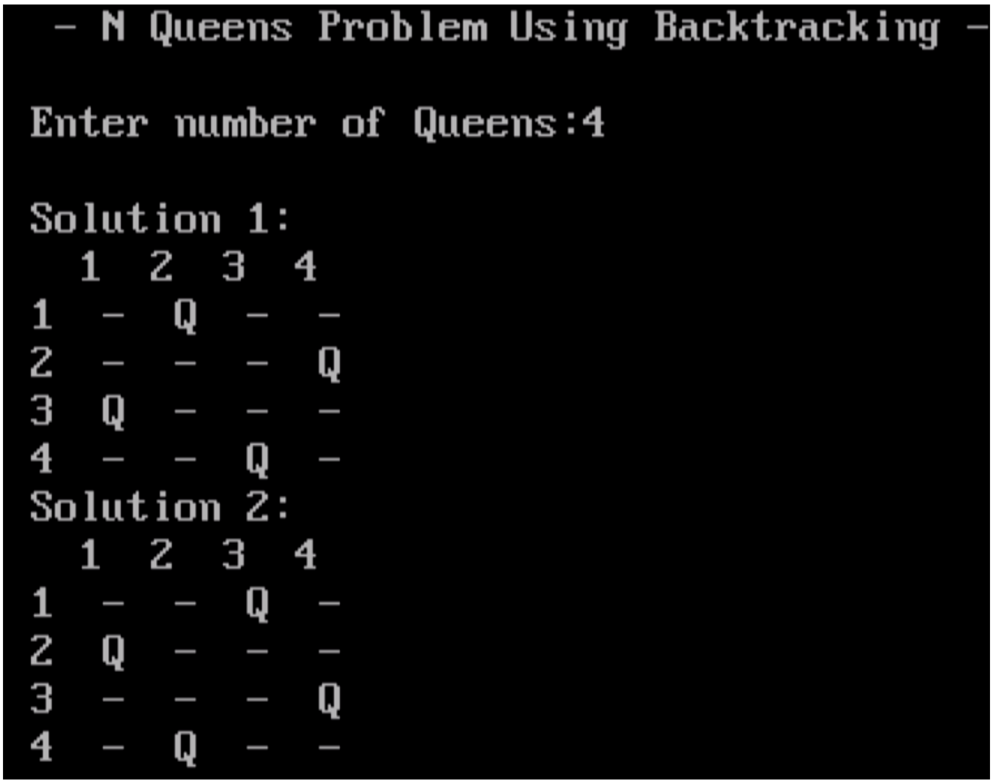
Program Code:

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
```

```
int board[20],count;

void queen(int row,int n);
int place(int row,int column);
void print(int n);

int main()
{
 int n,i,j;
 clrscr();


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

//function for printing the solution
void print(int n)
{
 int i,j;
 printf("\nSolution %d:\n",++count);

 for(i=1;i<=n;++i)
  printf("  %d",i);

 for(i=1;i<=n;++i)
 {
  printf("\n%d",i);
  for(j=1;j<=n;++j) //for nxn board
  {
   if(board[i]==j)
    printf("  Q"); //queen at i,j position
   else
    printf("  -"); //empty slot
  }
 }
}

/*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);
  }
 }
}
```

Output Snapshot:



```
  - 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  -  -
```

Application

Local search or constraint programming is useful to solve the problems like creating a solution which fulfills some condition.

Outcome:

Successfully implemented N queen problem using Backtracking Strategy in C.