FLAT ASSIGNMENT-2

M.AKSHARA

21071A6737

N-QUEENS PROBLEM

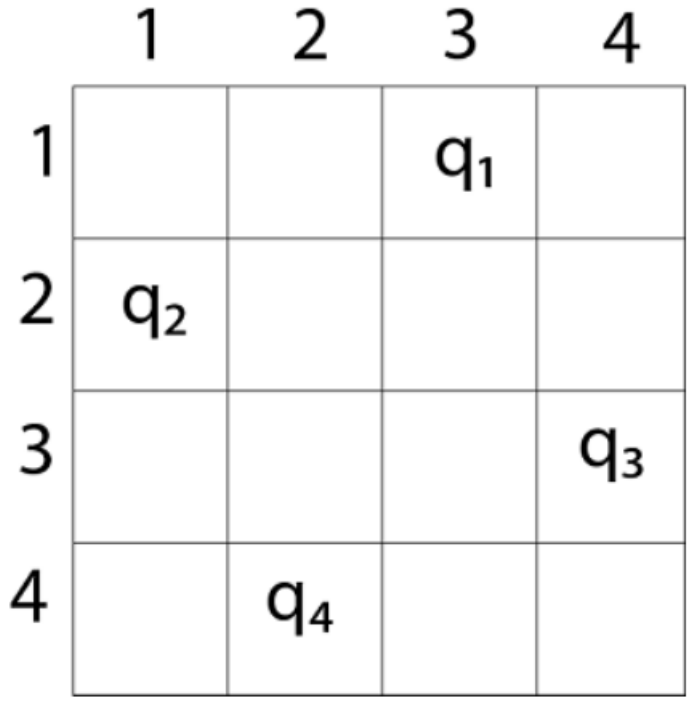
N - Queens problem is to place n - queens in such a manner on an n x n chessboard that no queens attack each other by being in the same row, column or diagonal.

It can be seen that for n =1, the problem has a trivial solution, and no solution exists for n =2 and n =3. So first we will consider the 4 queens problem and then generate it to n - queens problem.

Given a 4 x 4 chessboard and number the rows and column of the chessboard 1 through 4

Since, we have to place 4 queens such as q_1 q_2 q_3 and q_4 on the chessboard, such that no two queens attack each other. In such a conditional each queen must be placed on a different row, i.e., we put queen "i" on row "i."

Now, we place queen q_1 in the very first acceptable position (1, 1). Next, we put queen q_2 so that both these queens do not attack each other. We find that if we place q_2 in column 1 and 2, then the dead end is encountered. Thus the first acceptable position for q_2 in column 3, i.e. (2, 3) but then no position is left for placing queen 'q_3' safely. So we backtrack one step and place the queen 'q_2' in (2, 4), the next best possible solution. Then we obtain the position for placing 'q_3' which is (3, 2). But later this position also leads to a dead end, and no place is found where 'q_4' can be placed safely. Then we have to backtrack till 'q_1' and place it to (1, 2) and then all other queens are placed safely by moving q_2 to (2, 4), q_3 to (3, 1) and q_4 to (4, 3). That is, we get the solution (2, 4, 1, 3). This is one possible solution for the 4-queens problem. For another possible solution, the whole method is repeated for all partial solutions. The other solutions for 4 - queens problems is (3, 1, 4, 2) i.e.



SOURCE CODE

```
#include <stdio.h>
#include <stdlib.h>

int queenpos[50],flag;


 int place (int k,int i) {

 for(int j=1;j<k;j++) {

     if(queenpos[j] == i)
       return 0;

     if( j-k == queenpos[j]-i || k-j == queenpos[j]-i)
        return 0;
 }

 return 1;

 }

void Nqueen(int k,int n) {
```

```
  int i;

  for(i=1;i<=n;i++) {

      if(place(k,i) == 1) {

          queenpos[k] = i;

          if(k == n) {
              printf("Queen positions are \n");
               for(int i=1;i<=n;i++)
                printf("%d ",queenpos[i]);
               flag = 1;
                exit(7);
          }

          else
            Nqueen(k+1,n);

      }
  }

 }
int main()
{
   int n;
   printf("Enter number of queens u wanna use for NQUEEN \n");
   scanf("%d",&n);

   Nqueen(1,n);

  if (flag == 0)
   printf("Solution Does Not Exist \n");

   return 0;
}
```

OUTPUT:-

```
Enter number of queens u wanna use for NQUEEN
4
Queen positions are
2 4 1 3
Process returned 7 (0x7)   execution time : 10.356 s
Press any key to continue.
```

```
Enter number of queens u wanna use for NQUEEN
8
Queen positions are
1 5 8 6 3 7 2 4
Process returned 7 (0x7)   execution time : 5.102 s
Press any key to continue.
```