FLAT ASSIGNMENT 2

Ganesh Chaitanya B

21071A6716

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)

The **eight queens puzzle** is the problem of placing eight chess queens on an 8×8 chessboard so that no two queens threaten each other; thus, a solution requires that no two queens share the same row, column, or diagonal. There are 92 solutions. The problem was first posed in the mid-19th century. In the modern era, it is often used as an example problem for various computer programming techniques.

The eight queens puzzle is a special case of the more general ***n* queens problem** of placing n non-attacking queens on an $n \times n$ chessboard. Solutions exist for all natural numbers n with the exception of $n = 2$ and $n = 3$. Although the exact number of

solutions is only known for $n \leq 27$, the asymptotic growth rate of the number of solutions is approximately $(0.143\ n)^n$.

N-Queens problem:

```
#include <iostream>
#include <algorithm>
using namespace std;

bool safe(int** arr,int x,int y,int n)
{
	int row,col;
	for(row=0;row<x;row++)
	{
		if(arr[row][y]==1)
		{
			return false;
		}
	}
	row=x;
	col=y;
	while(row>=0&&col>=0)
	{
		if(arr[row][col]==1)
		{
			return false;
		}
		row--;
		col--;
```

```
        }
        row=x;
        col=y;
        while(row>=0&&col<n)
        {
                if(arr[row][col]==1)
                {
                        return false;
                }
                row--;
                col++;
        }
        return true;
}


bool nqueen(int** arr,int x,int n)
{
        int col;
        if(x>=n)
        {
                return true;
        }
        for(col=0;col<n;col++)
        {
                if(safe(arr,x,col,n))
                {
                        arr[x][col]=1;
```

```
                        if(nqueen(arr,x+1,n))
                        {
                                return true;
                        }
                        arr[x][col]=0;
                }
        }
        return false;
}

int main()
{
        int n,i,j;
        cout<<"Enter the value of n:"<<endl;
        cin>>n;
        int** arr=new int*[n];
        for(i=0;i<n;i++)
        {
                arr[i]=new int[n];
                for(j=0;j<n;j++)
                {
                        arr[i][j]=0;
                }
        }


        if(nqueen(arr,0,n))
```

```
    {
        for(i=0;i<n;i++)
        {
            for(j=0;j<n;j++)
            {
                cout<<arr[i][j]<<" ";
            }
            cout<<endl;
        }
    }
    return 0;
}
```

Output: