**Week 12: Write a java programs to implement backtracking algorithm for the N-queens problem.**

import java.io.\*;

class operation

{

int x[]=new int[20];

int count=0;

public boolean place(int row,int column)

{

int i;

for(i=1;i<=row-1;i++)

{ //checking for column and diagonal conflicts

if(x[i] == column)

return false;

else

if(Math.abs(x[i] - column) == Math.abs(i - row))

return false;

}

return true;

}

public void Queen(int row,int n)

{

int column;

for(column=1;column<=n;column++)

{

if(place(row,column))

{

x[row] = column;

if(row==n)

print\_board(n);//printing the board configuration

else //try next queen with next position

Queen(row+1,n);

}

}

}

public void print\_board(int n)

{

int i;

System.out.println("\n\nSolution :"+(++count));

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

{

System.out.print(" "+i);

}

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

{

System.out.print("\n\n"+i);

for(int j=1;j<=n;j++)// for nXn board

{

if(x[i]==j)

System.out.print(" Q");

else

System.out.print(" -");

}

}

}

}

class Backtrackqueen

{

public static void main (String args[] )throws IOException

{

DataInputStream in=new DataInputStream(System.in);

System.out.println("Enter no Of Queens");

int n=Integer.parseInt(in.readLine());

operation op=new operation();

op.Queen(1,n);

}

}