```
#include <iostream>
#include<cstdlib>
#include <cmath>

using namespace std;

bool ok(int q[], int col){
if  the configuration is “bad”  return false;
   else
   return true;
};

void backtrack(int &col){
   col--;
         if(col==-1) exit(1);
};

void print(int q[]){
   static int count =0;
   print the array q
};

 int main(){
   int q[8]; q[0]=0;
         int c=1;

   // from_backtrack keeps track if we need to reset the row to the
   // top of the current colum or not.
   bool from_backtrack=false;


   // The outer loop keeps looking for solutions
   // The program terminates from function backtrack
   // when we are forced to backtack into column -1
   while(1){
      while(c<8){ //this loop goes across columns

         // if we just returned from backtrack, use current value of row
         // otherwise get ready to start at the top of this column
         if(!from_backtrack) // we did not just return from backtrack
                  Code goes here

         from_backtrack=false;

         while(q[c]<8){ // place queen in this column

                  q[c]++;

            // if row=8, there is no valid square in this column
            // so backtrack and continue the loop in the previous column

                  Code goes here

            //if this position is ok, place the queen
            // and move on (break) to the next column,
            // otherwise keep looking in this column
            Code goes here
         }
         c++; // placed ok, move to the next column
      }
      // one complete solution found, print it.
      print(q); // board completed, print it out
      backtrack(c);
      from_backtrack=true;
   }
}
```