The 8 Queens Problem

We will solve the 8 Queens problem using BACKTRACKING. Actually we will solve the n Queens problem.

- What is BACKTRACKING?
- What is the 8 Queens problem?
- What is the n Queens problem?

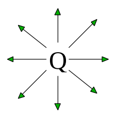
Backtracking

Backtracking is kind of solving a problem by trial and error. However, it is a well organized trial and error. We make sure that we never try the same thing twice. We also make sure that if the problem is finite we will eventually try all possibilities (assuming there is enough computing power to try all possibilities).

The 8 Queens Problem:

Given is a chess board. A chess board has 8x8 fields. Is it possible to place 8 queens on this board, so that no two queens can attack each other?

NOTES: A queen can attack horizontally, vertically, and on both diagonals, so it is pretty hard to place several queens on one board so that they don't attack each other.



The **n** Queens problem:

Given is a board of **n** by **n** squares. Is it possible to place **n** queens (that behave exactly like chess queens) on this board, without having any one of them attack any other queen?

Example: 2 Queens problem is not solvable.

Q	Q

Q	
Q	

Q	
	Q

	Q
Q	

Q	Q

	Q
	Q

Example 2: The 4-queens problem is solvable:

		Q	
Q			
			Q
	Q		

Basic idea of solution:

- Start with one queen in the first column, first row.
- Start with another queen in the second column, first row.
- Go down with the second queen until you reach a permissible situation.

- Advance to the next column, first row, and do the same thing.
- If you cannot find a permissible situation in one column and reach the bottom of it, then you have to go back to the previous column and move one position down there.
  (This is the backtracking step.)

- If you reach a permissible situation in the last column of the board, then the problem is solved.
- If you have to backtrack BEFORE the first column, then the problem is not solvable.

A slow example:

Q	Q	

illegal

Q		
	Q	

illegal

Q		
	Q	

legal

Q		Q
	Q	

illegal

Q		
		Q
	Q	

illegal

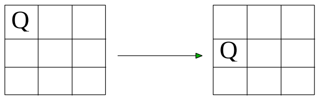
Q		
	Q	Q

illegal

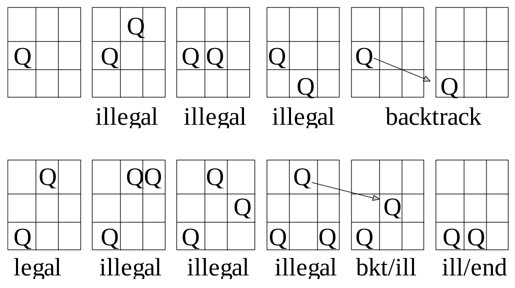
I cannot go further down in row 3.
I must backtrack!

Q		
	Q	

However, I cannot go further down in column 2 either. I must backtrack one more step.



Now I start again in the second column.



At this point I am at the end of the first column. I would have to backtrack again, but that's impossible, so the problem is unsolvable.

We will work out a successful example also:

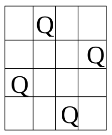
```
QQ++  Q+++  Q+++  Q+Q+  Q+++  Q+++
++++  +Q++  ++++  ++++  ++Q+  ++++
++++  ++++  +Q++  +Q++  +Q++  +QQ+
++++  ++++  ++++  ++++  ++++  ++++
illegal illegal legal illegal illegal illegal

Q+++  Q+Q+  Q+++  Q+++  Q+++
++++  ++++  ++Q+  ++++  ++++
+Q++  ++++  ++++  ++Q+  ++++
++Q+  +Q++  +Q++  +Q++  +QQ+
illegal illegal illegal illegal illegal
```

Backtrack to column 2. Then backtrack to column 1. Then go down in column 1.

```
+Q++  3       ++++  ++Q+  ++QQ  ++Q+  ++Q+
Q+++  ill     Q+++  Q+++  Q+++  Q++Q  Q+++
++++  steps   ++++  ++++  ++++  ++++  +++Q
++++          +Q++  +Q++  +Q++  +Q++  +Q++
illegal                   ill   ill   legal
```

I placed 4 queens on a 4x4 board.
Problem solved.

Are there other solutions? There must be, due to summetry:



We can continue the program to find this and other solutions.

The program is long and complicated. It certainly does not fit onto one screen. It moves from one row to the next row with a **while** loop. However, it moves from one column to the next column by recursion. This is done as follows:

```
void PlaceNextQueen (int Column,
                     boardtypes &Board,
                     boolean &Done)
{
  .............
  .............
     PlaceNextQueen(Column+1, Board, Done);
}
```

Please study the program in your textbook. Good exercise: Try to write the FUNCTION **Attack** which is used in the book, but not explained. **Attack**(Board, Row, Column) returns TRUE if on the board there is any queen that can attack the field in row Row and column Column.