CS110: Computer Programming Lab
Department of CSE
IIT, Guwahati

Module 04 Stage 01 Exercise 09

Exercise

A chessboard is a board game with play area made of 64 squares organised as 8 rows of 8 columns each. Thus, one can view it as a 2-dimensional array `char chess[8][8]`.

Queen is a versatile piece in the game of chess. A queen located at square `(c, r)` can attack all pieces on the chess board that are located in any of these four straight lines from the queen.

1. Same column as the queen. That is, square `(x, y)` is under attack if `x==c`.
2. Same row as the queen. That is, square `(x, y)` is under attack if `y == r`.
3. Same forward diagonal as the queen. That is, square `(x, y)` is under attack if `(y - x) == (r - c)` and both squares are on the chess-board. And,
4. Same backward diagonal as the queen. That is, square `(x, y)` is under attack if `(y + x) == (r + c)` and both squares are on the chess-board.

One can write a program with 4 recursive functions to place 8 queens in a chess-board. The functions are:

`int safeColumn (char chess[8][8], int r, int c)`
Returns 1 if square `(c, r)` is not attacked by any queen located in a `row < r`.

`int safeForwardDiag (char chess[8][8], int r, int c)`
Returns 1 if square `(c, r)` is not attacked by any queen located in a `row < r` along a forward diagonal. Similarly,

`int safeBackwardDiag (char chess[8][8], int r, int c)`
Returns 1 if square `(c, r)` is not attacked by any queen located in a `row < r` along a backward diagonal.

Thus, if we have already safely placed queens in rows `0` through `r-1`, one more queen can be placed in row `r` at location `(c, r)` if the abovementioned three functions all determine location `(c, r)` to be a safe location. The recursive function for doing this can be specified as:

`int insertQueens (char chess[8][8], int r)`
The function inserts queens in row `r` to row 7 recursively. It return the number of different ways these queens could be placed in these rows.

In this program you must compute the number of different ways one can insert 8 queens on a chess-board.

My implementation of the program, including function `main()`, has 61 lines (without comments but with about 10 blank spacing lines).

The program prints the final answer that is listed below.

```
Number of solutions = 92
```

This following part is speculative. Please report if it is wrong.

The following (recursive) identity may help you write this program to compute value `insertQueens(0)`:

`insertQueens(r)` = $\sum$ `insertQueens(r+1)` for all safe squares `(c, r)`

`insertQueens(7)` = Number of safe squares in row 7.
This will be either 0 or 1.

Unfortunately, the recurrence function are not easy to compute as the configurations of queens in rows `0` to `r-1` affects the values of `insertQueens(r+1)`.

An easy way to count the number of solutions in recursive function setting is as follows. When a solution is found at the end of a series of recursive calls, increment the count of solutions found. However, report it as unsuccessful to the calling recursive function. The calling function would then continue its efforts to find solution.

Error Reporting and Suggestions for Improvements

My sincere apologies if the document has errors or mistakes. Please report errors in this document to vmm@iitg.ernet.in. Also, I welcome suggestions and advice to improve the quality of the document for the students of CS110.