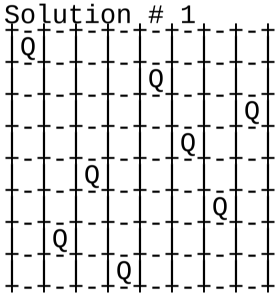
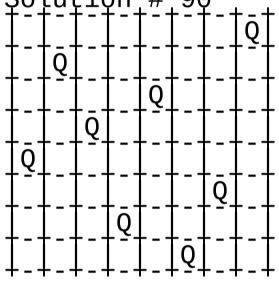
```
vip% cat queen.c
/* Queen's problem: locate N queens on an N-by-N chessboard,
 *    so that no two attack one another.
 * Solution by Neal Wagner, 4 April 1995
 */
#include <stdio.h>
#include <stdlib.h>
#define N 8
char b[N][N];  /* the chess board */
int row = -1;  /* row whose queen is currently placed */
int sol =  0;  /* number of solutions found */

void addqueen(void);
void writeboard(void);

void main(void)
{
   int row, c;
   for (row = 0; row < N; row++)
      for (c = 0; c < N; c++)
         b[row][c] = ' ';
   addqueen();
}

void addqueen(void)
{
   int c; /* column being tried for the queen */
   int i, level, j1, j2;  /* used to calculate attacks */
   int noattack;  /* 1 means no queen attacking */
   row++;
   for (c = 0; c < N; c++) {
      if (row == 0) { /* row 0, just place queen */
         b[row][c] = 'Q';
         addqueen();
      }
      else {  /* try queen in column c */
         noattack = 1;
         for (i = row - 1; i >= 0; i--){
            level = row - i;
            j1 = c - level;
            j2 = c + level;
            if ( (b[i][c]  == 'Q') ||
                 (j1 >= 0 && b[i][j1] == 'Q') ||
                 (j2 <  N && b[i][j2] == 'Q')) {
               noattack = 0;
               break;
            }
         }
         if (noattack) {
            b[row][c] = 'Q';
            if (row == N - 1) {
               sol++;
               getchar();
               printf("Solution # %d\n", sol);
               writeboard();
            }
            else
               addqueen();
         }
      }
      /* now try next location for queen */
      b[row][c] = ' '; /* in case it got set */
   } /* for */
   /* now need to back up a row */
   row--;
}

void writeboard(void)
{
   int i, j;
   for (i = 0; i < N; i++) {
      for (j = 0; j < N; j++)
         printf("+-");
      printf("+\n");
      for (j = 0; j < N; j++)
         printf("|%c", b[i][j]);
      printf("|\n");
   }
   for (j = 0; j < N; j++)
      printf("+-");
   printf("+\n");
}
```

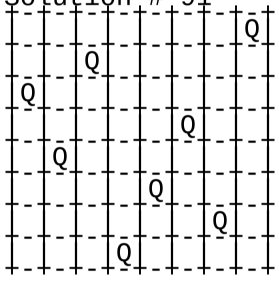
Solution # 1



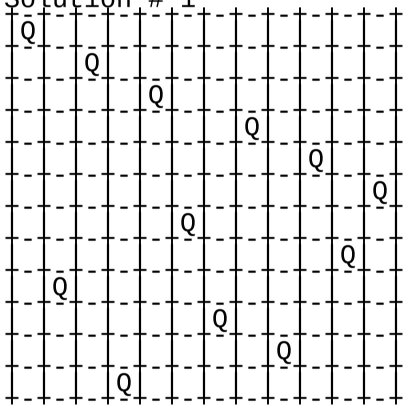
Solution # 90



Solution # 91



Solution # 1



Solution # 1