```
 1  /* Problem 4--Polyominoes Are Falling Down
 2     Since this game did not involve any sort of strategy, all you really
 3     have to do is see how down a piece falls. */
 4
 5  import java.io.*;
 6  import java.util.*;
 7
 8  public class prob4 {
 9
10   private static Scanner in;
11   private static PrintWriter out;
12   private static int cs;
13   private static int width;
14   private static Vector<String> board;
15
16   public static void main (String[] args) throws Exception {
17
18    in = new Scanner (new File ("prob4.in"));
19    out = new PrintWriter ("prob4.out");
20    cs = 1;
21    while (true) {
22     width = Integer.parseInt (in.nextLine());
23     if (width==0) break; //Read in the width
24     board = new Vector ();
25     while (true) { //Read in each piece
26      Vector<String> piece = ReadPiece ();
27      if (piece.isEmpty()) break;
28      int loc = BestFit (piece); //find out where the piece goes
29      Place (piece,loc);  //put it there
30      Shrink();  //get rid of lines
31     }
32     out.printf ("Case %d:\r\n",cs++); //print board
33     for (int i = board.size()-1; i>=0;i--)
34      out.printf ("%s\r\n",board.get(i));
35     out.printf ("\r\n");
36    }
37    in.close ();
38    out.close ();
39   }
40
41   /* This reads in a single piece */
42   public static Vector<String> ReadPiece () {
43
44    Vector<String> p = new Vector ();
45    while (true) {
46     String line = in.nextLine ();
47     if (line.equals("")) break;
48     p.add(0,line);
49    }
50    return p;
51   }
52
53   /* Compatible tests whether pieces overlap (are *'s overlapping) */
54   public static boolean Compatible (String a, String b) {
55
56    for (int i=0; i < a.length(); i++)
57     if (a.charAt(i)=='*' && b.charAt(i)=='*') return false;
58    return true;
59   }
60
61   /* Fit tests whether a certain piece would fit in a certain row.
62      Do all its rows steer clear of pieces already fallen? */
63   public static boolean Fit (Vector<String> p, int loc) {
64
65    int rows = Math.min (p.size(),board.size()-loc);
66    for (int i=0; i < rows; i++)
```

```
 67     if (!Compatible (p.get(i),board.get(i+loc))) return false;
 68    return true;
 69   }
 70
 71   /* Place puts the piece on the board at the designated location. */
 72   public static void Place (Vector<String> p, int loc) {
 73
 74    int rows = Math.min (p.size(),board.size()-loc);
 75    for (int i=0; i < rows; i++) {
 76     String l1 = p.get(i), l2 = board.get(i+loc);
 77     String ans = ""; //add the *'s
 78     for (int j=0; j < l1.length(); j++)
 79      if (l1.charAt(j)=='*') ans += '*';
 80      else ans += l2.charAt(j);
 81     board.set(i+loc,ans);
 82    } //Just copy over the top rows
 83    for (int i=rows; i < p.size(); i++)
 84     board.add (p.get(i));
 85   }
 86
 87   /* BestFit finds the best place to put the piece.  It starts above the
 88      board and moves down */
 89   public static int BestFit (Vector<String> p) {
 90
 91    for (int loc = board.size()-1; loc >= 0; loc--)
 92     if (!Fit(p,loc)) return loc+1;
 93    return 0;
 94   }
 95
 96   /* AllStars returns whether a row is made up entirely of *'s */
 97   public static boolean AllStars (String s) {
 98
 99    for (int i=0; i < s.length(); i++)
100     if (s.charAt(i) != '*') return false;
101    return true;
102   }
103
104   /* Shrink drops the All-Starred rows from the board. */
105   public static void Shrink () {
106
107    for (int i=0; i < board.size(); i++) {
108     String l = board.get(i);
109     if (AllStars (l)) {
110      board.remove (i);
111      i--;
112     }
113    }
114   }
115  }
116
```