```
 1  /* Problem 3--Always Look On The Bright Side Of Life
 2     The way I did it was I located LIFE and BRIGHT and treated the axis
 3     containing LINE as a linear inequality.  I tested which side of the
 4     inequality BRIGHT was on, and looked for a BRIAN that was on the
 5     other side. */
 6
 7  import java.io.*;
 8  import java.util.*;
 9
10  public class prob3 {
11
12   private static Scanner in;
13   private static PrintWriter out;
14   private static int cs, r, c;
15   private static char[][] Grid;
16
17   public static void main (String[] args) throws Exception {
18
19    in = new Scanner (new File ("prob3.in"));
20    out = new PrintWriter ("prob3.out");
21    cs = 1;
22    while (true) {
23     r = in.nextInt(); //Get row and col information
24     c = in.nextInt();
25     if (r==0 && c==0) break;
26     in.nextLine();
27     ReadIn (); //Get the word grid
28     Process (); //Solve the problem
29    }
30    in.close();
31    out.close();
32   }
33
34   /* Read in the grid a row at a time */
35   public static void ReadIn () throws Exception {
36
37    Grid = new char[r][c];
38    for (int i=0; i < r; i++) Grid[i] = in.nextLine().toCharArray();
39   }
40
41   /* Search for words and check for the BRIGHT side of LIFE. */
42   public static void Process () throws Exception {
43
44    Location[] LIFE = Find ("LIFE"); //Find all instances of LIFE, BRIGHT,
45    Location[] BRIGHT = Find ("BRIGHT"); //and BRIAN.  There had better be
46    Location[] BRIAN = Find ("BRIAN"); //only one LIFE and only one BRIGHT
47    int brightside = side (LIFE[0].getr(),LIFE[0].getc(),
48         LIFE[0].getr()+LIFE[0].getdr(),LIFE[0].getc()+LIFE[0].getdc(),
49         BRIGHT[0].getr(),BRIGHT[0].getc()); //which side of LIFE is
50    int i=0; //BRIGHT on, positive or negative?
51    for (;;i++) //Go through all the BRIANs until you get one whose side
52     if (brightside==side (LIFE[0].getr(),LIFE[0].getc(), //is brightside
53         LIFE[0].getr()+LIFE[0].getdr(),LIFE[0].getc()+LIFE[0].getdc(),
54         BRIAN[i].getr(),BRIAN[i].getc()) &&
55         brightside==(side (LIFE[0].getr(),LIFE[0].getc(),
56         LIFE[0].getr()+LIFE[0].getdr(),LIFE[0].getc()+LIFE[0].getdc(),
57         BRIAN[i].getr()+BRIAN[i].getdr()*4,
58         BRIAN[i].getc()+BRIAN[i].getdc()*4))) break;
59    for (int p = 0; p < 5; p++) //Lowercase all the letters of that BRIAN
60     Grid[BRIAN[i].getr()+BRIAN[i].getdr()*p]
61         [BRIAN[i].getc()+BRIAN[i].getdc()*p] += 32;
62    Print(); //Print the grid
63   }
64
```

```
 65   /* Just prints the new grid */
 66   public static void Print () throws Exception {
 67
 68    out.printf ("Case %d:\r\n\r\n",cs++);
 69    for (int i=0; i < r; i++)
 70     out.printf ("%s\r\n",new String(Grid[i]));
 71    out.printf ("\r\n");
 72   }
 73
 74   /* If you have two points (a,b),(c,d), the line going through them is
 75      given by (b-d)x + (c-a)y +ad-bc = 0.  Some (x,y) might make this
 76      formula positive and some might make it negative; that's how you
 77      know which side of the line you're on. */
 78   private static int side (int a, int b, int c, int d, int row, int col)
 79     throws Exception {
 80
 81    int comp = (b-d)*row+(c-a)*col + a*d-b*c;
 82    int side;
 83    if (comp > 0) side = 1;
 84    else if (comp < 0) side = -1;
 85    else side = 0;
 86    return side;
 87   }
 88
 89   /* Find locates all instances of a word in the grid and stores them in
 90      an array */
 91   public static Location[] Find (String word) throws Exception {
 92
 93    Location[] l = new Location[r*c];
 94    int lct = 0;
 95    for (int i = 0; i < r; i++)
 96     for (int j=0; j < c; j++)
 97      for (int dr=-1; dr <= 1; dr++) //Loop through all directions
 98       for (int dc=-1; dc <= 1; dc++) //horizontally vertically
 99        if (dr!=0 || dc != 0) { //diagonally
100         String thisword="";
101         for (int k=0; k < word.length(); k++)
102          thisword += G(i+k*dr,j+k*dc); //Build word
103         if (thisword.equals(word)) l[lct++] = new Location (i,j,dr,dc);
104        } //Did we find it?  Add location and direction to array
105    return l;
106   }
107
108   /* Easy way to look in the array.  It returns a space is we go out of
109      bounds...thus preventing us from going out of bounds! */
110   public static char G (int i, int j) throws Exception {
111
112    char letter;
113    if (i < 0 || i >=r || j < 0 || j >= c) letter = ' ';
114    else letter = Grid[i][j];
115    return letter;
116   }
117  }
118
119  /* Location stores the first letter of the word [r,c] and the direction
120     of the word [dr,dc].  Forward is [1,0].  Backward is [-1,0].
121     Northeast = [1,-1], and so forth. */
122  class Location {
123
124   private int r,c,dr,dc;
125
126   /* Accessors */
127   public int getr () throws Exception {return r;}
128   public int getc () throws Exception {return c;}
```

```
129   public int getdr () throws Exception {return dr;}
130   public int getdc () throws Exception {return dc;}
131
132   /* Constructor */
133   public Location (int R, int C, int DR, int DC) throws Exception {
134    r = R; c = C; dr = DR; dc = DC;
135   }
136  }
137
```