```java
/* Problem 6--Word Brick
   You know, there are 26 directions a word can move in, and you don't
   have to code any of them manually.  You can iterate through them with
   nested loops.
   A "direction" can be an ordered triple (dl,dw,dh) where all three are
   between -1 and 1 without all of them being 0. */

import java.io.*;
import java.util.*;

public class prob6 {

 private static Scanner in;
 private static PrintWriter out;
 private static int l, w, h, cs;
 private static char[][][] Brick;
 private static String word;

 public static void main (String[] args) throws Exception {

  in = new Scanner (new File ("prob6.in"));
  out = new PrintWriter ("prob6.out");
  cs = 1;
  while (true) {
   l = in.nextInt(); w = in.nextInt(); h = in.nextInt();
   if (l==0 && w==0 && h==0) break;
   in.nextLine();
   ReadBrick();
   Process();
  }
  in.close ();
  out.close ();
 }

 public static void ReadBrick() throws Exception {

  Brick = new char[l][w][h]; /* Read in this mess */
  for (int i=0; i < l; i++)
   for (int j=0; j < w; j++)
    Brick[i][j] = in.next().toCharArray();
  word = in.next();

 }

 public static void Process() throws Exception {

  int i=0, j=0, k=0, dl, dw, dh;

  outer:for (i=0; i < l; i++) /* outer three loops control position */
   for (j=0; j < w; j++) /* of starting letter */
    for (k=0; k < h; k++)
     for (dl = -1; dl <= 1; dl++) /* These three control direction */
      for (dw = -1; dw <= 1; dw++)
       inner:for (dh = -1; dh <= 1; dh++) { /* Skip (0,0,0) */
        if (dl==0 && dw==0 && dh==0) continue;
        for (int pos = 0; pos < word.length(); pos++)
         if (access (i+pos*dl,j+pos*dw,k+pos*dh)!=word.charAt(pos))
          continue inner; /* If it's no match, keep going */
        break outer; /* If match, end all loops NOW */
       }
  out.printf ("Case %d: ",cs++);
  if (i < l)
   out.printf ("%s was found at (%d,%d,%d)\r\n\r\n",word,i+1,j+1,k+1);
  else
   out.printf ("%s was not found.\r\n\r\n",word);
 }

 public static char access(int i, int j, int k) throws Exception {

   /* access just handles out-of-bound checking */
  if (i < 0 || i >= l || j < 0 || j >= w || k < 0 || k >= h)
   return (char)0; /* If out of bounds return null byte */
  return Brick[i][j][k]; /* which will guarantee no match */
 }
}
```