```
 1  /* Problem 1--Internal Palindromes
 2     One of the longest common subsequences between a string and its
 3     reverse must be a palindrome, so we find the longest such
 4     subsequences and print those that are palindromes. */
 5
 6  import java.io.*;
 7  import java.util.*;
 8
 9  public class prob1 {
10
11   private static Scanner in = null;
12   private static PrintWriter out = null;
13   private static int cs=0;
14
15   public static void main (String[] args) throws Exception {
16
17    in = new Scanner (new File ("prob1.in"));
18    out = new PrintWriter ("prob1.out");
19
20    while (true) {
21     String line = in.nextLine(); //read input until empty line
22     if (line.equals ("")) break;
23     Process (line);
24    }
25    in.close ();
26    out.close ();
27   }
28
29   /* Process processes each string and prints the palindromes */
30   public static void Process (String line) {
31
32    String[][][] info = new String[line.length()+1][line.length()+1][];
33    //Storing arrays of strings for various subsequences
34
35    String enil = "";
36    for (int i=0; i < line.length(); i++) enil = line.charAt(i)+enil;
37    //Reverse the string
38
39    for (int i=0; i < info.length; i++)
40     for (int j=0; j < info[i].length; j++) {
41      //Everyone starts with a maximum subsequence of ""
42      info[i][j] = new String[1]; info[i][j][0] = "";
43      if (i==0 || j==0) continue;
44      if (line.charAt(i-1)==enil.charAt(j-1)) {
45       //We can build longer strings by attaching the same character
46       info[i][j] = new String[info[i-1][j-1].length];
47       for (int k=0; k < info[i][j].length; k++)
48        info[i][j][k] = info[i-1][j-1][k]+line.charAt(i-1);
49      }
50      //Merge in strings of equal length
51      if (info[i-1][j][0].length() > info[i][j][0].length())
52       info[i][j] = info[i-1][j];
53      else if (info[i-1][j][0].length() == info[i][j][0].length())
54       info[i][j] = Merge (info[i-1][j],info[i][j]);
55      if (info[i][j-1][0].length() > info[i][j][0].length())
56       info[i][j] = info[i][j-1];
57      else if (info[i][j-1][0].length() == info[i][j][0].length())
58       info[i][j] = Merge (info[i][j-1],info[i][j]);
59     }
60    out.println ("Case "+(++cs)+":");
61    out.println (); //Print those strings that are palindromes
62    for (int i=0; i < info[line.length()][line.length()].length; i++)
63     PrintPal (info[line.length()][line.length()][i]);
64    out.println ();
65   }
66
67   /* Merge merges two sorted lists of strings, deleting duplicates */
68   public static String[] Merge (String[] A, String[] B) {
```

```
69
70    String[] C = new String[A.length+B.length];
71    int i=0, j=0, ct=0;
72    while (i < A.length || j < B.length) {
73     if (i==A.length) C[ct++] = B[j++];
74     else if (j==B.length) C[ct++] = A[i++];
75     else if (A[i].equals(B[j])) {C[ct++]=A[i++];j++;}
76     else if (A[i].compareTo (B[j]) < 0) C[ct++]=A[i++];
77     else C[ct++] = B[j++];
78    }
79    String[] M = new String[ct];
80    for (i=0; i < ct; i++) M[i] = C[i];
81    return M;
82   }
83
84   /* Prints a string if it's a palindrome */
85   public static void PrintPal (String A) {
86
87    String B = "";
88    for (int i=0; i < A.length(); i++) B = A.charAt(i)+B;
89    if (A.equals(B)) out.println (A);
90   }
91
92  }
93
```