```
 1  /* Problem 1--Roman Roads
 2     This was a greedy algorithm problem.  Some watering holes are already
    covered by feed stores.  For the others, start at the left and ONLY
    place a feed store when you absolutely have to.  This will end up being
    the minimum number of feed stores. */
 3
 4  import java.io.*;
 5  import java.util.*;
 6
 7  public class prob1 {
 8
 9   private static Scanner in;
10   private static PrintWriter out;
11   private static int cs;
12   private static int N, D, M;
13   private static int[] cp, cpc;
14   private static int ncp;
15
16   public static void main (String[] args) throws Exception {
17
18    in = new Scanner (new File ("prob1.in"));
19    out = new PrintWriter ("prob1.out");
20    cs = 1;
21    while (true) {
22     N = in.nextInt(); //Number of watering holes
23     if (N==0) break;
24     ReadIn (); //Read in the input
25     Process (); //Process the input
26    }
27    in.close ();
28    out.close ();
29   }
30
31   //Reads in the data case
32   public static void ReadIn () {
33
34    D = in.nextInt(); //Maximum distance from watering hole
35    cp = new int[N]; //Specific watering holes
36    for (int i=0; i < N; i++) cp[i] = in.nextInt();
37    M = in.nextInt (); //Number of feed stores
38    cpc = new int[M]; //Specific feed stores
39    for (int i=0; i < M; i++) cpc[i] = in.nextInt();
40   }
41
42   //Processes and evaluates the data case
43   public static void Process () {
44
45    ncp = 0; //number of new feed stores
46    int j=0; //j goes through the watering holes
47    for (int i=0; i <= M; i++) { //i goes
48  //           through (and beyond) the feed stores
49      while (j < N && (i==M || cp[j] < cpc[i]-D)) {
50       int p = j; //If we're not covered by the next feed store
51       while (p < N && cp[j] >= cp[p]-D) p++;
52        //Move forward until we found a water hole not covering hole j
53       ncp++;  //Make the previous watering hole a feed store
54       j = p;
55       while (j < N && cp[j] <= cp[p-1]+D) j++;
56         //Move forward until no longer covered
57      }
58      while (j < N && cp[j] <= cpc[i]+D) j++;
59         //Move forward until no longer covered
60    }
61    out.printf (
```

```
62     "Case %d: %d new feed store(s) will need to be added.\r\n\r\n",
63      cs++,ncp);
64   }
65  }
66
```