```
 1  /* Problem 1--Perfect Numbers
 2    This was a straightforward iteration of factor sums.  You only need to
 3    check up to the square root to find all factors. */
 4
 5  import java.io.*;
 6  import java.util.*;
 7
 8  public class prob1 {
 9
10   private static Scanner in;
11   private static PrintWriter out;
12   private static int cs;
13
14   public static void main (String[] args) throws Exception {
15
16    in = new Scanner (new File ("prob1.in"));
17    out = new PrintWriter ("prob1.out");
18    cs = 1;
19    while (true) { //read in the values
20     int r = in.nextInt ();
21     int k = in.nextInt ();
22     int f = in.nextInt ();
23     int l = in.nextInt ();
24     if (r==0 && k==0 && f==0 && l==0) break;
25     Process (f,l,r,k);
26    }
27    in.close ();
28    out.close ();
29   }
30
31   //This iterates the Factor Sum r times and checks whether the answer is
32   //k times the original number
33   public static boolean IsPerfect (int r, int k, int n)
34    throws Exception {
35
36    int p = n;
37    for (int i=0; i < r; i++) p = FS (p);
38    return p == k*n;
39   }
40
41   //Processes each data case
42   public static void Process (int f, int l, int r, int k)
43    throws Exception {
44
45    out.print ("Case "+(cs++)+":\r\n\r\n");
46    out.print ("The ("+r+","+k+")-perfect numbers between "+f+" and "+l+
47               " are:\r\n");
48    for (int i=f; i <= l; i++)
49     if (IsPerfect (r,k,i)) out.print (i+"\r\n");
50    out.print ("\r\n");
51   }
52
53   //Computes the Factor Sum of n by checking all factors up to the square
54   //root.
55   public static int FS (int n) throws Exception {
56
57    int sum = 0;
58    for (int i = 1; i*i <= n; i++)
59     if (n%i==0) { //If the number is a perfect square, only count the
60      sum += i;    //square root once.
61      if (n/i > i) sum += n/i;
62     }
63    return sum;
64   }
```

```
65  }
66
```