import java.io.\*;

public class Problem4

{

public static void main(String[] args) throws IOException

{

FileReader file = new FileReader("DATA41.txt");

BufferedReader read = new BufferedReader(file);

for (int m = 0; m < 10; m++)

{

String line = read.readLine();

String input[] = line.split(" ");

long points[] = new long[2];

for (int x = 0; x < 2; x++)

points[x] = Long.parseLong(input[x]);

int loops = 9999; // this is how many fibonacci number squares the program will generate. I don't know if/how you make an array without a fixed size, so I just made it a really big number, lmao

long fib[][] = new long[loops][3];

long x = 0;

long y = 0;

int d = 0;

for (int i = 0; i < loops; i++)

{

String out = ""; // anything with "out" are just optional comments I used to fix some bugs while I was making the program work

if (i < 2)

{

fib[i][0] = 1;

}

else

{

fib[i][0] = fib[i-1][0] + fib[i-2][0];

}

out += fib[i][0];

x = direction(fib, i, d, true); // true indicates that I'm calculating the change in x position, false is the change in y position

y = direction(fib, i, d, false);

fib[i][1] = x;

fib[i][2] = y;

out += "\tx: " + fib[i][1];

out += "\ty: " + fib[i][2];

out += "\ts: " + fib[i][0];

//System.out.println(out);

if (fallsUnder(fib, i, points[0], points[1]))

{

System.out.println(i + 1);

break;

}

d++;

if (d > 3)

d = 0;

}

} // close master loop

read.close();

}

public static long direction(long[][] fib, int i, int d, boolean isX) // to find which direction and how far the square's point of origin will move relative to the last one. this method works for x and y movevements, but only when run separately twice

{

long size = fib[i][0];

long ls = 0; // last size (from the square before it)

long os = 0; // old size (two squares before the current one)

long x = 0;

long y = 0;

if (i > 0) // prevents extending beyond the array

ls = fib[i-1][0];

if (i > 1)

os = fib[i-2][0];

if (i > 0)

{

x = fib[i-1][1];

y = fib[i-1][2];

}

if (d==0) // down

{

x -= os;

y -= ls;

}

else if (d==1) // left

{

x -= size;

y += os;

}

else if (d==2) // up

{

y += size;

}

else if (d==3) // right

{

x += ls;

}

if (isX)

return x;

else

return y;

}

public static boolean fallsUnder(long fib[][], int i, long xA, long yA) // xA and yA are the values of the point we are evaluating to see whether or not they full into the fibonacci square

{

boolean isUnder = false;

long size = fib[i][0];

long x = fib[i][1];

long y = fib[i][2];

//System.out.println("Check " + i + ". Does (" + xA + ", " + yA + ") fall under (" + x + ", " + y + ") with a size of " + size + "?");

//if (xA >= x) System.out.println("TRUE1"); if (xA <= (x + size)) System.out.println("TRUE2"); if (yA >= y) System.out.println("TRUE3"); if (yA <= (y + size)) System.out.println("TRUE4");

if ((xA >= x) && (yA <= y) && (xA <= (x + size)) && (yA >= (y - size)))

isUnder = true;

return isUnder;

}

}