题目描述：

某探险队负责对地下洞穴进行探险。探险队成员在进行探险任务时，随身携带的记录器会不定期地记录自身的坐标，但在记录的间隙中也会记录其他数据。探索工作结束后，探险队需要获取到某成员在探险过程中相对于探险队总部的最远的足迹位置。

1. 仪器记录坐标时，坐标的数据格式为(x,y)，如(1,2)、(100,200)，其中 0<x<1000，0<y<1000。同时存在非法坐标，如(01,1)、(1,01)，(0,100)属于非法坐标。

2. 设定探险队总部的坐标为(0,0)，某位置相对总部的距离为：x*x+y*y。

3. 若两个座标的相对总部的距离相同，则第一次到达的坐标为最远的足迹。

4. 若记录仪中的坐标都不合法，输出总部坐标（0,0）。

备注：不需要考虑双层括号嵌套的情况，比如 sfsdfsd((1,2))。

输入描述：

字符串，表示记录仪中的数据。

如：ferga13fdsf3(100,200)f2r3rfasf(300,400)

输出描述：

字符串，表示最远足迹到达的坐标。

如： (300,400)

示例 1

输入：

```
ferg(3,10)a13fdsf3(3,4)f2r3rfasf(5,10)
```

输出：

```
(5,10)
```

说明：

记录仪中的合法坐标有 3 个： (3,10)， (3,4)， (5,10)，其中(5,10)是相距总部最远的坐标， 输出(5,10)。

示例 2

输入：

```
asfefaweawfaw(0,1)fe
```

输出：

```
(0,0)
```

说明：

记录仪中的坐标都不合法，输出总部坐标（0,0）

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Main {
    public static void main(String[] args) throws Exception {
        Pattern pattern = Pattern.compile("\\((\\d+),(\\d+)\\)");
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        while (true) {
            String line = br.readLine();
            if (line == null) {
                break;
            }
            int maxDis = 0;
            String maxCor = "(0,0)";
            Matcher matcher = pattern.matcher(line);
            while (matcher.find()) {
                String sx = matcher.group(1);
                String sy = matcher.group(2);
                int x;
                int y;
                try {
                    x = Integer.parseInt(sx);
                    y = Integer.parseInt(sy);
                } catch (NumberFormatException e) {
                    continue;
                }
```

```
                    if (x <= 0 || x >= 1000) {
                        continue;
                    }
                    if (y <= 0 || y >= 1000) {
                        continue;
                    }
                    if (!Integer.toString(x).equals(sx)) {
                        continue;
                    }
                    if (!Integer.toString(y).equals(sy)) {
                        continue;
                    }
                    if (x * x + y * y > maxDis) {
                        maxDis = x * x + y * y;
                        maxCor = matcher.group(0);
                    }
                }
                System.out.println(maxCor);
            }
        }
    }
```