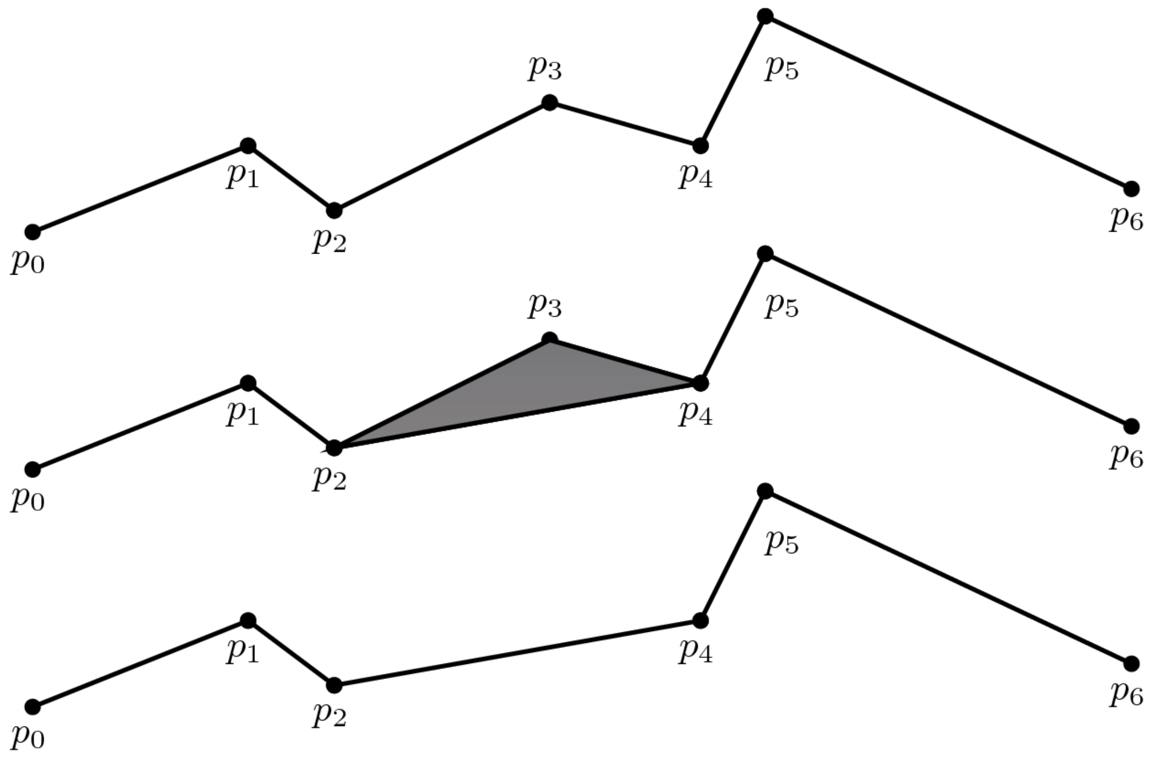
Problem C. Polyline Simplification

Time limit	3000 ms
Mem limit	1048576 kB
OS	Linux

Mapping applications often represent the boundaries of countries, cities, etc. as polylines, which are connected sequences of line segments. Since fine details have to be shown when the user zooms into the map, these polylines often contain a very large number of segments. When the user zooms out, however, these fine details are not important and it is wasteful to process and draw the polylines with so many segments. In this problem, we consider a particular polyline simplification algorithm designed to approximate the original polyline with a polyline with fewer segments.

A polyline with n segments is described by $n + 1$ points $p_0 = (x_0, y_0), \ldots, p_n = (x_n, y_n)$, with the ith line segment being $\overline{p_{i-1}p_i}$. The polyline can be simplified by removing an interior point p_i $(1 \le i \le n-1)$, so that the line segments $\overline{p_{i-1}p_i}$ and $\overline{p_ip_{i+1}}$ are replaced by the line segment $\overline{p_{i-1}p_{i+1}}$. To select the point to be removed, we examine the area of the triangle formed by p_{i-1}, p_i , and p_{i+1} (the area is 0 if the three points are colinear), and choose the point p_i such that the area of the triangle is smallest. Ties are broken by choosing the point with the lowest index. This can be applied again to the resulting polyline, until the desired number m of line segments is reached.

Consider the example below.



The original polyline is shown at the top. The area of the triangle formed by p_2, p_3, and p_4 is considered (middle), and p_3 is removed if the area is the smallest among all such triangles. The resulting polyline after p_3 is removed is shown at the bottom.

Input

The first line of input contains two integers n ($2 \leq n \leq 200\,000$) and m ($1 \leq m < n$). The next $n + 1$ lines specify $p_0, \ldots, p_n$. Each point is given by its x and y coordinates which are integers between -5000 and 5000 inclusive. You may assume that the given points are strictly increasing in lexicographical order. That is, $x_i < x_{i+1}$, or $x_i = x_{i+1}$ and $y_i < y_{i+1}$ for all $0 \leq i < n$.

Output

Print on the kth line the index of the point removed in the kth step of the algorithm described above (use the index in the original polyline).

Sample 1

Input	Output

```
10 7
0 0
1 10
2 20
25 17
32 19
33 5
40 10
50 13
65 27
75 22
85 17
```
	```
	1
	9
	6
	```