Problem J. Closest Pair Algorithm

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	512 mebibytes

There is a classic problem about finding two closest points amongst a set of points on a plane. There is also a classic randomized *algorithm* to solve it, which goes like this:

```
rotate the plane around the origin by a random angle phi
let p be the array of points
sort p by x coordinate in increasing order
ans = INF;
for (int i = 0; i < n; i++) {
    for (int j = i - 1; j >= 0; j--) {
        if (p[i].x - p[j].x >= ans) break;
        ans = min(ans, dist(p[i], p[j]));
    }
}
```

Here "INF" is some number greater than all distances. The function "dist" returns Euclidean distance between the given points. Note that all x coordinates are distinct after rotation with probability 1.

I know that the algorithm works well in practice, but how well exactly? I ask you to compute the expected number of calls of the function "dist", assuming that the angle "phi" is chosen uniformly at random from the range $[0; 2 \cdot \pi)$.

Input

The first line contains a single integer n ($2 \leq n \leq 250$) — the number of points.

The next n lines contains coordinates of points, one per line.

All the coordinates are not greater than 10^6 by absolute value.

It is guaranteed that all points are distinct. It is guaranteed that no three points lie on the same line.

Output

Print one number — the expected number of calls of the function "dist".

Your answer is considered correct if its absolute or relative error does not exceed 10^{-6}.

Formally, let your answer be a, and the jury's answer be b. Your answer is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \leq 10^{-6}$.

Examples

standard input	standard output
4 0 0 0 1 1 0 1 1	5.0000000000000
3 0 0 0 1 1 0	2.5000000000000