#include<stdio.h>

#include<math.h>

typedef struct point

{

float x;

float y;

float angle;

}Point;

int checkCCW(Point a, Point b, Point c)

{

float res;

res = ((a.x \* b.y) + (b.x \* c.y) + (c.x \* a.y)) - ((c.y \* a.x) + (a.y \* b.x) + (b.y \* c.x));

if (res < 0)

return -1;

else

return 1;

}

void initialPoint(Point p[], int length)

{

int i;

for (i = 0; i < length; i++)

{

if (i == 0)

{

p[i].x = 0;

p[i].y = 0;

continue;

}

scanf("%f %f", &p[i].x, &p[i].y);

}

}

float ComputeAngle(Point A, Point B)

{

int Dx, Dy;

float angle;

Dx = B.x - A.x;

Dy = B.y - A.y;

if ((Dx >= 0) && (Dy == 0))

angle = 0;

else

{

angle = fabs((float)(Dy)) / (fabs((float)(Dx)) + fabs((float)(Dy)));

if ((Dx < 0) && (Dy >= 0))

angle = 2 - angle;

else if ((Dx <= 0) && (Dy < 0))

angle = 2 + angle;

else if ((Dx>0) && (Dy < 0))

angle = 4 - angle;

}

return angle \* 90.0;

}

void sort(Point p[], int length)

{

int i,j;

Point temp;

for (i = 0; i < length; i++)

{

for (j = i; j < length; j++)

{

if (p[i].angle > p[j].angle)

{

temp = p[i];

p[i] = p[j];

p[j] = temp;

}

}

}

}

float calculate(Point stack[], int count)

{

int i;

float res = 0;

for (i = 0; i < count - 1; i++)

{

res += sqrt((float)(((stack[i].x - stack[i + 1].x) \* (stack[i].x - stack[i + 1].x)) + ((stack[i].y - stack[i + 1].y) \* (stack[i].y - stack[i + 1].y))));

}

res += sqrt((float)(((stack[0].x - stack[count - 1].x) \* (stack[0].x - stack[count - 1].x)) + ((stack[0].y - stack[count - 1].y) \* (stack[0].y - stack[count - 1].y))));

return res;

}

void Graham(Point p[], Point stack[], int length)

{

int i;

int count = 0;

for (i = 0; i < length; i++)

{

p[i].angle = ComputeAngle(p[0], p[i]);

}

sort(p, length);

for (i = 0; i < length; i++)

{

if (i == 0 || i == 1)

{

stack[count] = p[i];

count++;

}

else

{

if (checkCCW(stack[count - 2], stack[count - 1], p[i]) == 1)

{

stack[count] = p[i];

count++;

}

else

{

stack[count - 1] = p[i];

}

}

}

printf("result : %.2f\n", calculate(stack, count) + 2);

}

int main(void)

{

Point \* p;

Point \* stack;

int length;

scanf("%d", &length);

length++;

p = (Point\*)malloc(sizeof(Point) \* length);

stack = (Point\*)malloc(sizeof(Point) \* length);

initialPoint(p, length);

Graham(p,stack, length);

}