Решение:

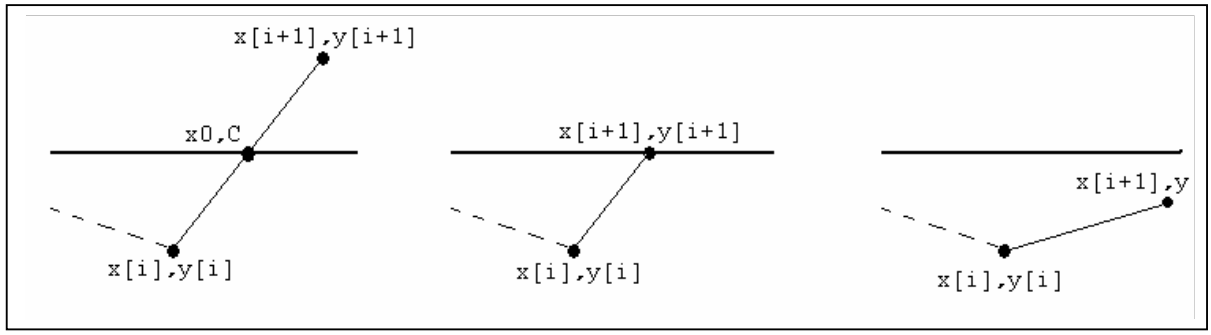
Както в повечето задачи на изчислителната геометрия, и в тази задача може да се забележи наличието на много различни случаи, които трябва да бъдат разгледани. Пропускането на някой от случаите би могъл да доведе до неточно решение. Затова трябва да бъдат пълно и точно класифицирани всички случаи. Един начин да направим това е с помощта на краен автомат. Без да влизаме в подробности (желаещите да се запознаят по-подробно с понятието могат да го направят в кой да е учебник по Дискретна математика) ще кажем, че крайните автомати, които ще използваме, се определят с множеството Q от *състоянията* си, множеството V от *входовете* си, множеството W от *изходите* си и правило, по което от текущото състояние q и вход v се определя следващото състояние q' и изхода w.

Ще разгледаме автомат с две състояния: 0 – поредният връх на полигона се намира над или върху разделителната линия $y = C$ и тази част от полигона не участва в пресмятаното лице; 1 – поредният връх на полигона се намира под разделителната линия и тази част от полигона участва в пресмятаното лице. В началото автоматът е в състояние 0. Входовете на автомата ще бъдат 9 (кодирани с числата от 0 до 8) и се определят от положението на краищата на текущата отсечка спрямо разделителната линия. Ако левият край е над разделителната линия, то входът ще бъде 0, 1 или 2 в зависимост от това дали десният край а над, върху или под разделителната линия. Аналогично, случаите с ляв край на текущата отсечка върху разделителната права кодираме с 3, 4 или 5, а случаите с ляв край на текущата отсечка под разделителната права – с 6, 7 или 8. Функцията `calc_input`, по зададен в променливата `i` номер на поредната отсечка на полигона, пресмята в променливата `j` съответната стойност на входа, а в променливата `x0` пресмята, ако е необхдимо, x-координатата на пресечната точка на отсечката с разделителната линия.

В таблицата е дефиниран съответният автомат, като за всяко от състоянията и всеки от входовете са покзани следващото състояние и (след знака „;”) съответният изход. Отсъствието на изход е показано с „–”.

	вх. 0	вх. 1	вх. 2	вх. 3	вх. 4	вх. 5	вх. 6	вх. 7	Вх. 8
съст. 0	0; –	0; –	1; w_1	0; –	0; –	1; w_2	0; w_3	0; w_4	1; w_5
съст. 1	1; –	1; –	1; –	1; –	1; –	1; –	0; w_6	0; w_7	1; w_8

Изходите на автомата въщност са кодът който трябва да се изпълни, при съответно стояние и вход. Те се състоят от добавяне в масива `p` (с индекс `L`) на координатите на една или повече точки и евентуално пресмятане (с функцията `calc_face`) на лицето на поредното „парче”, ако такова парче вее се е формирало. Ще разгледаме по-подробно дефиницията на автомата в състояние 1 и ще оставим на читателя за упражнение дефиницията в състояние 0. Ако се намираме в състояние 1, т.е. под разделителната линия, тогава входовете 0, 1 и 2 (левият край на отсечката е над разделителната линия) и входовете 3, 4 и 5 (левият край на отсечката е върху разделителната линия) са невъзможни. Затова можем да считаме, че автомат остава в същото състяние и не е нужно дефинирането на изход. За случаите 6, 7 и 8 виж Фигурата



Във всички случаи точката с координати (`x[i],y[i]`) вече е в масива `p`, затова в случай 6 (w_6) поставяме в него и точката с координати (`x0,C`) . Това завършва поредното парче. Добавяме в края и началната точката на парчето с координати (`p[X][0],p[Y][0]`) и пресмятаме лицето му. Тъй като краят на отсечката е над разделителнта линия, сменяме

състоянието на 0. Случаят 7 (w_7) е аналогичен, но вместо (x0,C) поставяме в масива p координатите (x[i+1],y[i+1]). В случая 8 (w_8) поставяме в масива p само координатите (x[i+1],y[i+1]). Те не завършват парче, затова не пресмятаме лице, а тъй като краят на отсечката остава под линията, не сменяме и състоянието.

```
#include <stdio.h>
#define X 0
#define Y 1
#define MAXN 2005
int N; double C; double x[MAXN],y[MAXN],p[2][MAXN];
void calc_input(int i,int* j,double* x0)
{    int a,b; double d;
     if(y[i]>C) a=0;
     else {if (y[i]==C) a=1; else a=2;}
     if(y[i+1]>C) b=0;
     else {if (y[i+1]==C) b=1; else b=2;}
     *j=3*a+b;
     if(*j==2||*j==6)
     {
        if(x[i]==x[i+1]) *x0=x[i];
        else
        {
           d=(x[i+1]-x[i])/(y[i+1]-y[i]);
           *x0=x[i]+(C-y[i])*d;
        }
     }
}
double calc_face(int L)
{  double s=0.;int i;
   for(i=1;i<=L;i++)
      s+=(p[Y][i]+p[Y][i-1])*(p[X][i]-p[X][i-1]);
   if(s<0) s= -s;
   return s/2.;
}
main()
{
    int i,j,L=0,state=0; double face=0.,x0,y0;
    scanf("%d %lf",&N,&C);
    for(i=0;i<=N;i++) scanf("%lf %lf",&x[i],&y[i]);
    for(i=0;i<N;i++)
    {  calc_input(i,&j,&x0);
       if(state==0)
           switch(j)
           {   case 2: p[X][L]=x0;p[Y][L++]=C;
                       p[X][L]=x[i+1];p[Y][L++]=y[i+1];
                       state=1;
                       break;
               case 5: p[X][L]=x[i];p[Y][L++]=y[i];
                       p[X][L]=x[i+1];p[Y][L++]=y[i+1];
                       state=1;
                       break;
```

```
                case 6: p[X][L]=x[i];p[Y][L++]=C;
                        p[X][L]=x[i];p[Y][L++]=y[i];
                        p[X][L]=x0;p[Y][L++]=C;
                        p[X][L]=x[i];p[Y][L]=C;
                        face+=calc_face(L);L=0;
                        break;
                case 7: p[X][L]=x[i];p[Y][L++]=C;
                        p[X][L]=x[i];p[Y][L++]=y[i];
                        p[X][L]=x[i+1];p[Y][L++]=y[i+1];
                        p[X][L]=x[i];p[Y][L]=C;
                        face+=calc_face(L);L=0;
                        break;
                case 8: p[X][L]=x[i];p[Y][L++]=C;
                        p[X][L]=x[i];p[Y][L++]=y[i];
                        p[X][L]=x[i+1];p[Y][L++]=y[i+1];
                        state=1;
                        break;
            }
        else
            switch(j)
            {
                case 6: p[X][L]=x0;p[Y][L++]=C;
                        p[X][L]=p[X][0];p[Y][L]=p[Y][0];
                        face+=calc_face(L);L=0;
                        state=0;
                        break;
                case 7: p[X][L]=x[i+1];p[Y][L++]=y[i+1];
                        p[X][L]=p[X][0];p[Y][L]=p[Y][0];
                        face+=calc_face(L);L=0;
                        state=0;
                        break;
                case 8: p[X][L]=x[i+1];p[Y][L++]=y[i+1];
                        break;
            }
    }
    if(state==1)
    {
       p[X][L]=x[N];p[Y][L++]=C;
       p[X][L]=p[X][0];p[Y][L]=p[Y][0];
       face+=calc_face(L);
    }
    printf("%lf\n",face);
    return 0;
}
```

Задача B1. СТЕПЕН

Напишете програма **POWER**, която въвежда от един ред на стандартния вход три цели числа n, k и p ($10^4 < n < 10^9$, $0 < k < 10^9$, $0 < p < 5$) и извежда на стандартния изход последните p цифри на n^k.

ПРИМЕР
Вход
```
1001 4 3
```
Изход
```
001
```