**A**

Після екзамену n магістрів зайшли до ресторану відзначити цю непересічну подію. Директор ресторану запропону­вав їм і надалі відвідувати саме цей ресторан цього дня щороку, причому кожного разу сідати за той самий КРУГЛИЙ стіл іншим способом. Директор пообіцяв, що, після того як усі способи посад­ки за стіл будуть вичерпані, їх годуватимуть у ресторані безкош­товно. Коли настане цей день?

Input format

Натуральне число n≤1000

Output format

Рядок з цілим числом, яке містить точну відповідь.

Examples

|  |  |
| --- | --- |
| Input in input.txt | Output in output.txt |
| 3 | 6 |
|  |  |

#include "fstream"

using namespace std;

ifstream cin("input.txt");

ofstream cout("output.txt");

int main ()

{

int i,k,os;

int a[10001],n,c[10001];

for (i=0;i<10001;i++)a[i]=0;

cin>>n;

a[0]=1;a[1]=1;

for(k=1;k<=n;k++)

{

c[0]=a[0];

os=0;

for(i=1;i<=c[0];i++)

{

c[i]=(a[i]\*k+os)%10;

os=(a[i]\*k+os)/10;

}

while (os>0){c[0]=c[0]+1;c[c[0]]=os%10;os=os/10;}

for (i=0;i<=c[0];i++)a[i]=c[i];

}

for (i=c[0];i>=1;i--)cout<<c[i];

cout<<"\n";

return 0;

}

B

Написати програму для знаходження кількості N-розрядних чисел, що володіють наступними властивостями: - складаються лише з цифр 1, 2 та 3; - не містять підряд двох однакових цифр 1 та 3; - містять цифру 2 або групу з цифр 2 лише тоді, коли в записі числа зліва знаходиться цифра 1, а справа – цифра 3. Наприклад, при N=4 це будуть числа: 1313, 3131, 1231, 1223, 3123.

Input format

Єдиний рядок вхідного файлу містить натуральне число N (1≤N≤500).

Output format

Єдиний рядок вихідного файлу повинен містити одне натуральне число – відповідь.

Examples

|  |  |
| --- | --- |
| Input in input.txt | Output in output.txt |
| 4 | 5 |

// FIBONACHI5.cpp: определяет точку входа для консольного приложения.

//

#include<fstream>

#include<vector>

#include<string>

using namespace std;

#define base (long long)1e+9

typedef vector<long long> mas;

mas a,b;

int n;

ifstream f1("input.txt");

FILE \*f2=fopen("output.txt","w");

int sum(mas &a,mas &b)

{

 long long c=0;

 for(int i=0;i<max(a.size(),b.size()) || c;i++)

 {

 if(i==a.size()) a.push\_back(0);

 a[i]+=c+(i<b.size()?b[i]:0);

 c=a[i]>=base;

 if(c) a[i]-=base;

 }

 return 0;

}

int mult(mas &a,mas &b,mas &c)

{

 c.resize(a.size()+b.size());

 for(int i=0;i<a.size();i++)

 for(int j=0,x=0;j<b.size() || x;j++)

 {

 long long cr=c[i+j]+a[i]\*(j<b.size()?b[j]:0)+x;

 c[i+j]=cr%base;

 x=cr/base;

 }

 while(c.size()>1 && !c.back()) c.pop\_back();

 return 0;

}

int write(mas &a)

{

 fprintf(f2,"%d",a.empty()?0:a.back());

 for(int i=(int)a.size()-2;i>=0;i--)

 fprintf(f2,"%09d",a[i]);

 fprintf(f2,"\n");

 return 0;

}

void fib(int n)

{

 mas a11,a12,a21,a22,r11,r12,q11,q12,q21,q22,q;

 a11.push\_back(1);

 a12.push\_back(1);

 a21.push\_back(1);

 a22.push\_back(0);

 r11.push\_back(1);

 r12.push\_back(0);

 while(n)

 {

 if(n&1)

 {

 q11.clear();

 q12.clear();

 q.clear();

 mult(r11,a11,q11);

 mult(r12,a21,q);

 sum(q11,q);

 q.clear();

 mult(r11,a12,q12);

 mult(r12,a22,q);

 sum(q12,q);

 r11=q11;

 r12=q12;

 if(n==1) break;

 }

 q.clear();

 q11.clear();

 q12.clear();

 q21.clear();

 q22.clear();

 mult(a11,a11,q11);

 mult(a12,a21,q);

 sum(q11,q);

 q.clear();

 mult(a11,a12,q12);

 mult(a12,a22,q);

 sum(q12,q);

 q.clear();

 mult(a21,a11,q21);

 mult(a22,a21,q);

 sum(q21,q);

 q.clear();

 mult(a21,a12,q22);

 mult(a22,a22,q);

 sum(q22,q);

 a11=q11;

 a12=q12;

 a21=q21;

 a22=q22;

 n>>=1;

 }

 write(r12);

}

int main()

{

 f1>>n;

n++;

 fib(n);

}

С

Нехай n - довільне натуральне число, а послідовність i1, i2, ... , in містить усі натуральні числа від 1 до n включно. Порушенням порядку у такій послідовності називають систему таких двох нерівностей, що справджуються: j менше k та ij більше ik. Якщо послідовність зростає, то кількість порушень порядку дорівнює 0. Якщо послідовність спадає, то така кількість дорівнює n(n - 1)/2. В усіх інших випадках ця кількість розташована між вказаними величинами. Завдання Встановіть парність кількості порушень порядку послідовності.

Input format

У першому рядку вхідного файлу вказано кількість послідовностей m. Кожний з наступних m рядків містить натуральне число n і послідовність різних натуральних чисел від 1 до n включно: i1, i2, ... , in при 2 ≤ т ≤ 100, 2 ≤ n ≤ 1 048 576. У 50 % тестів n ≤ 4096.

Output format

Єдиний рядок вихідного файлу має містити число в шістнадцятковій системі числення, яке відповідає двійковому числу яке утворене з m символів - нулів або одиниць - без пропусків: k-й символ рядка - це залишок від ділення на 2 кількості порушень порядку k-ї послідовності, заданої (k + 1)-м рядком вхідного файлу.

Examples

|  |  |
| --- | --- |
| Input in input.txt | Output in output.txt |
| 53 1 2 33 2 3 13 1 3 24 2 3 4 14 3 4 1 2 | 6 |

#include "iostream"

#include "fstream"

using namespace std;

ifstream inp ("input.txt");

ofstream out ("output.txt");

int main()

{

 int a[104857],b[110];

char c[110];

int j,n,k,m,l,x,i;

 k=0;l=4;

 for(i=0;i<=110;i++)b[i]=0;

inp>>m;

 for (x=1;x<=m;x++)

 {

 inp>>n;

 for(i=1;i<=n;i++)inp>>a[i];

 //for(i=1;i<=n;i++)cout<<a[i]<<" ";cout<<"\n";

 k=0;

 for (j=1;j<=n-1;j++)

 for(i=j+1;i<=n;i++)if(a[j]>a[i])k++;

 l++;

 b[l]=k%2;

 }

//for(i=1;i<=l;i++)cout<<b[i];

//перевести двійкове число в 16

 i=l;x=0;

while (i>4)

{k=b[i]+b[i-1]\*10+b[i-2]\*100+b[i-3]\*1000;

x++;

if (k==0) c[x]='0';

if (k==1) c[x]='1';

if (k==10) c[x]='2';

if (k==11) c[x]='3';

if (k==100) c[x]='4';

if (k==101) c[x]='5';

if (k==110) c[x]='6';

if (k==111) c[x]='7';

if (k==1000) c[x]='8';

if (k==1001) c[x]='9';

if (k==1010) c[x]='A';

if (k==1011) c[x]='B';

if (k==1100) c[x]='C';

if (k==1101) c[x]='D';

if (k==1110) c[x]='E';

if (k==1111) c[x]='F';

i=i-4;

}

// відкинути 0 на початку

i=x;

while(c[i]=='0')i--;

for (j=i;j>=1;j--)

out<<c[j];

out<<"\n";

}

D

Відсортуйте N заданих чисел у неспадаючому порядку.

Input format

Задано число N (1 ≤ N ≤ 100000), а потім в одному чи декількох рядках N натуральних чисел з діапазону від -100 до 100.

Output format

Виведіть у стовпчик N чисел у неспадаючому порядку.

Examples

|  |  |
| --- | --- |
| Input in input.txt | Output in output.txt |
| 53 1 24 2 | 12234 |

// Швидке.cpp: определяет точку входа для консольного приложения.

//

#include <fstream>

using namespace std;

ifstream cin("input.txt");

ofstream cout("output.txt");

int a[1000];

int findpivot (int i, int j)

{

 int firstkey=a[i], f=0;

 for (int k=i+1;k<=j;k++)

 if (a[k]>firstkey)

 f=k;

 else if (a[k]<firstkey)

 f=i;

 return f;

}

int partition (int i, int j, int pivot)

{

 int l=i, r=j;

 do

 {

 int t=a[l];

 a[l]=a[r];

 a[r]=t;

 while (a[l]<pivot) l++;

 while (a[r]>=pivot) r--;

 } while (l<=r);

 return l;

}

void QuikSort (int i, int j)

{

 int pivot, pivotindex, k;

 pivotindex=findpivot(i, j);

 if (pivotindex!=0)

 {

 pivot=a[pivotindex];

 k=partition (i, j, pivot);

 QuikSort(i, k-1);

 QuikSort(k, j);

 }

}

int main()

{

 //a-вхiдний, b-вихiдний

 int n;

 int i;

 /\* нумерацiя починається з нуля\*/

 cin>>n;

 for (i=1; i<=n; i++) cin>>a[i];

 // сортування масиву. i-номер етапу, k-лiчильник елементiв

 QuikSort (1, n);

for (i=1; i<=n; i++)cout<<a[i]<<"\n";

 return 0;

}

E

За заданим числом N виведіть усі перестановки чисел від 1 до N у лексикографічному порядку.

Input format

Задано одне число N (0 менше N менше 10).

Output format

Необхідно вивести усі перестановки чисел від 1 до N у лексикографічному порядку. Перестановки виводяться по одній у рядку, числа у перестановці виводяться без пропусків.

Examples

|  |  |
| --- | --- |
| Input in input.txt | Output in output.txt |
| 3 | 123132213231312321 |

#include <iostream>

#include <fstream>

#include <math.h>

#include <vector>

#include <algorithm>

using namespace std;

vector <int> a;

 ifstream f;

 ofstream g;

void printper(int n);

int main()

{

 f.open("input.txt");

 g.open("output.txt");

 int n;

 f >> n;

 for (int i=1;i<=n;i++){

 a.push\_back(i);

 }

 printper(n);

 while (next\_permutation(a.begin(),a.end())){

 printper(n);

 };

 //printper(n);

 f.close();

 g.close();

 return 0;

}

void printper(int n)

{

 for (int i=0;i<n-1;i++){

 g << a[i];

 }

 g << a[n-1] << endl;

}

F

Під час досліджень, присвячених появі життя на планеті Олімпія, вченими було зроблено декілька сенсаційних відкриттів: 1. Усі живі організми планети походять від однієї бактерії Bitozoria Programulis. 2. Еволюція проходила крок за кроком (за припущенням вчених – під час змін клімату на планеті). 3. На кожному кроці еволюції з кожного виду утворювалися рівно два підвиди, а попередній вид зникав. 4. Якщо вважати появу бактерії Bitozoria Programulis першим кроком еволюції, то нині існуючі живі організми знаходяться на N-му кроці. Щоб не вигадувати назви під час досліджень, вчені пронумерували всі види організмів, що будь-коли існували на планеті. Для цього вони намалювали дерево еволюції із коренем Bitozoria Programulis, яка отримала номер 1. Далі нумерували види кожного кроку еволюції зліва направо. Таким чином безпосередні підвиди Bitozoria Programulis отримали номери 2 та 3. Наступними були занумеровані види третього кроку еволюції – підвиди виду 2 отримали номери 4 та 5, а виду 3 – номери 6 та 7, і т.д. Завдання Напишіть програму EVO, яка за номерами двох видів обчислить номер виду їх найближчого спільного предка у дереві еволюції.

Input format

Перший рядок вхідного файлу EVO.DAT містить ціле число N (1≤N≤100) – кількість етапів еволюції, що відбулися на планеті Олімпія до теперішнього часу. Другий та третій рядки файлу містять по одному натуральному числу, що представляють номери видів, для яких потрібно знайти номер їх найближчого спільного предка.

Output format

Єдиний рядок вихідного файлу EVO.SOL має містити натуральне число – номер найближчого предка для двох видів.

Examples

|  |  |
| --- | --- |
| Input in input.txt | Output in output.txt |
| 41512 | 3 |
| 18233016233008 | 14563 |

/\* gcc\_cpp \*/

#include <fstream>

#include <string>

using namespace std;

const char \*infile="evo.dat";

const char \*outfile="evo.sol";

inline int numcmp(const string &a,const string &b) {

 int asz=a.size(),bsz=b.size(),i;

 if(asz<bsz)return -1;

 else if(asz>bsz)return 1;

 for(i=0;i<asz&&a[i]==b[i];i++);

 if(i>=asz)return 0;

 else return a[i]-b[i];

}

inline string numdiv(const string &a,int d) {

 string res;

 int i,rem,asz=a.size();

 for(i=0,rem=0;i<asz;i++) {

 rem=10\*rem+a[i]-'0';

 if(rem/d||!res.empty())res.push\_back('0'+rem/d);

 rem%=d;

 }

 if(res.empty())res.push\_back('0');

 return res;

}

int main() {

 int n;

 string a,b;

 int c;

 ifstream ifs(infile);

 ofstream ofs(outfile);

 ifs>>n>>a>>b;

 while(c=numcmp(a,b)) {

 if(c>0)a=numdiv(a,2);

 else b=numdiv(b,2);

 }

 ofs<<a<<endl;

 return 0;

}

G

Турбаза мала для ночівлі N місць, з’єднаних стежками. Туристів можна вести в одну сторону. Довжина стежки – одноденний перехід. Пройти і перевірити всі M-денні маршрути, які починаються на базі K.

Input format

В першому рядку задано N,M,K. В наступних N рядках матриця сумыжності.

Output format

Одне число, яке визначає кількість маршрутів.

Examples

|  |  |
| --- | --- |
| Input in input.txt | Output in output.txt |
| 6 3 10 1 1 0 1 00 0 1 1 1 00 0 0 0 0 00 0 0 0 0 11 0 0 1 0 00 1 0 0 0 0 | 7 |

// graph1.cpp:

#include "fstream"

using namespace std;

ifstream cin("input.txt");

ofstream cout("output.txt");

int m,n,k;

 int c[100][100];

int np,im,kol;

int rez[100],stack[100];

void outresalt()

{int i;

//for(i=0;i<=m;i++) cout<<stack[i]+1<<" ";cout<<endl;

kol=kol+1;

}

int main()

{

 int i,j;

cin>>n>>m>>k;

 for(i=0;i<n;i++) {

for(j=0;j<n;j++)cin>>

 c[i][j];

 }

stack[0]=k-1;im=1;np=0;

while (im>0){

 while (np<=n && c[stack[im-1]][np]==0)np++;

 if (np>n-1) {im=im-1; np=stack[im]+1;}

 else {

 stack[im]=np;

 im=im+1;

 np=0;

 if (im-1==m){outresalt();im=im-1;np=stack[im]+1;}

 }

}

cout<<kol<<endl;

 return 0;

}

H

Зада́ча комівояже́ра (комівояжер — бродячий торговець; англ. Travelling Salesman Problem, TSP; нім. Problem des Handlungsreisenden) полягає у знаходженні найвигіднішого маршруту, що проходить через вказані міста хоча б по одному разу. В умовах завдання вказуються критерій вигідності маршруту (найкоротший, найдешевший, сукупний критерій тощо) і відповідні матриці відстаней, вартості тощо. Зазвичай задано, що маршрут повинен проходити через кожне місто тільки один раз, в такому випадку розв'язок знаходиться серед гамільтонових циклів.

Input format

Перший рядок N. Граф задається матрицею суміжності.

Output format

Довжина шляху.

Examples

|  |  |
| --- | --- |
| Input in input.txt | Output in output.txt |
| 30 1 21 0 32 3 0 | 6 |

// Komivojger1.cpp: определяет точку входа для консольного приложения.

//

#include <cstdlib>

#include <fstream>

using namespace std;

ifstream cin("input.txt");

ofstream cout("output.txt");

int main()

{

int a[200][200];

int min=1000000;

int s;

int n,x[100],y[100];

 int i,j,k,temp,found;

cin>>n;

for (i=1;i<=n;i++)

for (j=1;j<=n;j++)

{cin>>a[i][j];

}

for(i=1;i<=n;i++)x[i]=i;

 s=0;x[n+1]=x[1];

 for(j=1;j<=n;j++)s=s+a[x[j]][x[j+1]];

 if (s<min) {min=s; for(int m=1;m<=n;m++)y[m]=x[m];}

found=0;

while (found==0) {

 i=n;

found=0;

while (x[i-1]>=x[i]&&i>1) i=i-1;

if(i>1)k=i-1; else found=1;

if (found==0) {

 i=n;

 while (x[i]<=x[k])i=i-1;

 temp=x[i];

 x[i]=x[k];

 x[k]=temp;

 i=k+1;

 j=n;

 while (i<j){

 temp=x[j];

 x[j]=x[i];

 x[i]=temp;

 i=i+1;

 j=j-1; }

 s=0;x[n+1]=x[1];

 for(j=1;j<=n;j++)s=s+a[x[j]][x[j+1]];

 if (s<min) {min=s; for(int m=1;m<=n;m++)y[m]=x[m];}

 }

 }

// for(j=1;j<=n;j++)cout<<y[j]<<" "; cout<<"\n";

cout<<min<<"\n";

 return 0;

}

I

Король країни Аріїв завоював N міст на території сусідніх держав. Тепер йому необхідно створити систему збирання мита з завойованих територій. Він хоче збудувати таку систему шляхів між цими містами, щоб до будь-якого міста можна було дістатися (можливо, через інші міста) зі столиці, але у воєнному стані на транспорт виділяється дуже незначна частина фінансів, тому сумарна вартість побудованих шляхів сполучення між містами має бути мінімальною.

Input format

Перший рядок вхідного файлу містить натуральне число N (1≤N≤100) – кількість міст у країні, а також цілі числа X та Y – координати столиці. Наступні N рядків містять через проміжок координати Xi , Yi завойованих міст. Значення координат по модулю менші 50000.

Output format

Єдиний рядок має містити дійсне число з трьома знаками після коми – сумарну вартість побудованих доріг. Вважайте, що вартість одиниці довжини дороги дорівнює одній умовній одиниці.

Examples

|  |  |
| --- | --- |
| Input in input.txt | Output in output.txt |
| 6 0 0 1 1-1 1 0 2 1 -1-1 -1 0 -2 | 8.485 |
|  |  |

// Floid.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include <iostream>

#include <math.h>

using namespace std;

int main()

{const int n=25;

const int p=0;

int a[n][n];

int x[1000],y[1000],kol\_ver[1000],v[1000];

int k,i,j,vi,vj,min,s;

int ver[1000][3];

int f;

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

for (j=0;j<n;j++)

cin>>a[i][j];

k=0; v[k]=p;s=0;

while (k<n-1) {

min=100000;

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

for(j=0;j<n;j++)

if (a[v[i]][j]<min) {min=a[v[i]][j];vi=v[i];vj=j;}

f=1;

for (i=0;i<=k;i++)if (vj==v[i])f=0;

if (f==1) {k=k+1;

ver[k][1]=vi;

ver[k][2]=vj;

v[k]=vj;

kol\_ver[vj]=kol\_ver[vj]+1;

kol\_ver[vi]=kol\_ver[vi]+1;

s=s+a[vi][vj];

}

a[vi][vj]=1e30;a[vj][vi]=1e30;

}

cout<<s<<endl;

for(i=1;i<n;i++) cout<<ver[i][1]<<' '<<ver[i][2]<<endl;

return 0;

}

J

Нехай є n міст, пронумерованих числами від 1 до n. Для кожної пари міст із номерами і, j у таблиці a[і][j] зберігається ціле число - ціна прямого авіаквитка з міста i у місто j. Вважається, що рейси існують між будь-якими містами, a[і,і] = 0 при всіх і, a[і][j] може відрізнятися від a[j,і]. Найменшою вартістю проїзду з і в j вважається мінімально можлива сума цін квитків для маршрутів (у тому числі з пересадженнями), що ведуть з і в j. (Вона не перевершує a[і][j], але може бути менше).

Input format

N - кількість міст, a - старт, b - фініш. Матриця суміжності.

Output format

Найкоротша відстань.

Examples

|  |  |
| --- | --- |
| Input in input.txt | Output in output.txt |
| 3 1 20 37 5637 0 9956 99 0 | 37 |

#include<set>

#include <fstream>

using namespace std;

ifstream cin("input.txt");

ofstream cout("output.txt");

int st,fin, n;

int a[100][100];

int main()

{

int i,j,k,min;

cin>>n;

cin>>st;

cin>>fin;

st--;

fin--;

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

for(j=0;j<n;j++)

{cin>>a[i][j];

}

int dist[100],from[100];

set <int> s;

for(i=0;i<n;i++)s.insert(i);

s.erase(st); //{Підготовка множини не відвіданих вершин.}

for(i= 0;i<n;i++) //{Перенесення у масив dist інформації про відстань до вершин.}

{

 //{видимих зі стартової вершини st}

if (a[st][i]==0) dist[i]=65535; //Якщо ребро відсутнє, то відстань безмежна,}

else dist[i]= a[st][i]; //інакше така, як у таблиці суміжності. }

from[i]= st; //{Усі вершини графа видимі зі стартової вершинні}

}

from[st]= 0; dist[st]= 0; // {Стартова вершина видима ні з якої вершини.}

while (s.size()>0){

 //Виконання алгоритму, поки не будуть переглянуті всі вершини графа}

//{Визначення початкового мінімального значенню}

min = 65535; //{відстані між вершинами }

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

 //Перегляд усіх поточних відстаней між розглянутими вершинами}

if (s.find(i) != s.end() && dist[i] < min && dist[i] > 0) // {Визначення вершини k з}

 { min = dist[i]; k= i; }

 //{мінімальним поточним значенням відстані.}

for (i= 0;i<n;i++) //{Перегляд усіх вершин графа}

if (a[k, i] > 0 && dist[i] > dist[k] + a[k][i]) // {і визначення найкоротшого}

{

dist[i] = dist[k] + a[k][i]; //{поточного шляху між вершинами i та j.}

from[i]= k; //{Запам'ятовування номера вершини, через яку зроблено}

} //{перерахунок відстані.}

s.erase(k);// {Надання вершині k статусу "відвіданої".}

};

cout<<dist[fin]<<endl;

 return 0;

}

K

У школі вирішили організувати змагання спортивного орієнтування. В парку, де мало відбутися змагання, відмітили N контрольних пунктів, які потрібно пройти. Організатори журі вирішили обгородити територію. Але з метою економії коштів потрібно визначити спосіб побудови огорожі мінімальної довжини і площі, яка б охопила всі контрольні точки. Необхідно за заданими координатами точок обчислити площу території і довжину огорожі.

Input format

Вхідні дані: В першому рядку вхідного файлу записано N - кількість контрольних точок, в наступних рядках координати - (хi, уi), і=1,2,...,N. Всі числа цілі та знаходяться в межах від -1000 до 1000.

Output format

Вихідні дані: Містить два рядки з дійсними числами з точністю два знаки після коми, які відповідають площі території і довжині огорожі.

Examples

|  |  |
| --- | --- |
| Input in input.txt | Output in output.txt |
| 30 00 34 0 | 6.0012.00 |

#include <iostream>

#include <cstdio>

#include <vector>

#include <cmath>

using namespace std;

struct point

{

 int x,y;

 point(){}

 point(int X, int Y)

 {

 x = X;

 y = Y;

 }

};

bool operator != (const point &a, const point &b)

{

 return !(a.x == b.x && a.y == b.y);

}

double dist (const point &a, const point &b)

{

 return sqrt( 0.0 + (a.x - b.x) \* (a.x - b.x) + (a.y - b.y) \* (a.y - b.y));

}

int n;

vector<point> mas;

vector<int> convex\_hull;

double P,s;

void input()

{

 cin>>n;

 mas.resize(n);

 for (int i=0;i<n;i++)

 scanf("%d %d", &mas[i].x, &mas[i].y);

}

int OrientTriangl2(const point &p1,const point &p2, const point &p3)

{

 return p1.x \* (p2.y - p3.y) + p2.x \* (p3.y - p1.y) + p3.x \* (p1.y - p2.y);

}

bool isInside(const point &p1, const point &p, const point &p2)

{

 return ( p1.x <= p.x && p.x <= p2.x &&

 p1.y <= p.y && p.y <= p2.y);

}

void ConvexHullJarvis(const vector<point> &mas, vector<int> &convex\_hull)

{

 int base = 0;

 for (int i=1;i<n;i++)

 {

 if (mas[i].y < mas[base].y)

 base = i;

 else

 if (mas[i].y == mas[base].y &&

 mas[i].x < mas[base].x)

 base = i;

 }

 convex\_hull.push\_back(base);

 int first = base;

 int cur = base;

 do

 {

 int next = (cur + 1) % n;

 for (int i=0;i<n;i++)

 {

 int sign = OrientTriangl2(mas[cur], mas[next], mas[i]);

 if (sign < 0)

 next = i;

 else if (sign == 0)

 {

 if (isInside(mas[cur],mas[next],mas[i]))

 next = i;

 }

 }

 cur = next;

 convex\_hull.push\_back(next);

 }

 while (cur != first);

}

void solve()

{

 ConvexHullJarvis(mas,convex\_hull);

 for (size\_t i=0;i<convex\_hull.size()-1;i++)

 P += dist(mas[convex\_hull[i]],mas[convex\_hull[i+1]]);

 for (size\_t i=0;i<convex\_hull.size()-1;i++)

 s += mas[convex\_hull[i]].x\*mas[convex\_hull[i+1]].y-mas[convex\_hull[i+1]].x\*mas[convex\_hull[i]].y;

 s=0.5\*fabs(s);

}

void output()

{

 printf("%0.2f\n" ,s);

 printf("%0.2f\n",P);

///

}

int main()

{

 freopen("input.txt","r",stdin);

 freopen("output.txt","w",stdout);

 input();

 solve();

 output();

 return 0;

}