

Задача 1 (8 баллов) id 5136

Условие

Группа школьников из класса “К”, изучая работу с электронными таблицами захотела написать собственную программу по обработке входящей таблицы. Обработка должна заключаться в применении формулы, которая идёт вместе с таблицей.

На вход первой строкой поступает формула, результатом которой может быть любое целое число. Формула может иметь стандартные перечисленные математические знаки: +, -, *. Очерёдность операций определяется правилами математики, скобки не используются в формулах.

Далее идёт массив, представляющий из себя таблицу, где по горизонтали идут английские буквы от А до Е, по вертикали идут номера строк от 0 до 4. Программа получает на вход только числа, номера столбцов и строк в примере написаны для простоты понимания. Массив всегда имеет одинаковый размер.

Формула имеет следующий формат. В качестве элементов могут быть значения таблицы, к которым обращаются с помощью указания столбца и строки друг за другом, без пробела, например “A4”, “D2”.

На выходе программа должна вывести результат в виде целого числа, результат работы представленной формулы с данной таблицей.

Примечание: формула имеет от 2 до 5 слагаемых.

Пример: имеется формула и таблица (двумерный массив):

A2*C0-E3+C3

@	A	B	C	D	E
0	12	3	-10	67	8
1	34	5	2	34	11
2	3	5	-20	6	22

3 2 2 9 4 33

4 5 78 98 9 12

Ячейка A2 имеет значение 3. Ячейка C0 имеет зн

ачение -10. Ячейка E3 имеет значение 33. Ячейка C3 имеет значение 9. Получаем: $3 \cdot 10 - 33 + 9 = -54$

Входные данные:	Выходные данные:
$A2 \cdot C0 - E3 + C3$ 12 3 -10 67 8 34 5 2 34 11 3 5 -20 6 22 2 2 9 4 33 5 78 98 9 12	-54
$A0 \cdot D4 + B3 - A3 \cdot D0$ 24 -38 42 -12 -46 36 49 24 12 -48 43 -26 29 24 45 50 20 -45 -19 29 -2 16 -22 -50 -10	-580

Проверочные тесты

Входные данные	Ожидаемый результат
$A2 \cdot C0 - E3 + C3$ 12 3 -10 67 8 34 5 2 34 11 3 5 -20 6 22 2 2 9 4 33 5 78 98 9 12	-54
$A0 \cdot D4 + B3 - A3 \cdot D0$ 24 -38 42 -12 -46 36 49 24 12 -48 43 -26 29 24 45 50 20 -45 -19 29 -2 16 -22 -50 -10	-580
$D3 - A2 \cdot E3 + C3$ -44 -46 40 -49 -18 29 -38 12 44 49 -11 -46 -45 8 28 -33 18 -45 38 -39 8 28 28 9 -23	-436
$A0 \cdot A0 - C3 + B3$ 26 -1 35 34 -7 13 -10 29 -24 -43 -48 -5 46 -22 -25 35 20 -5 -7 43 26 -40 15 26 -47	701
$A0 \cdot D4 + B3$ -3 -20 -27 -37 22 -44 -42 -43 4 50 11 46 39 -17 -3 -40 -10 40 -19 -1 -18 0 -23 43 -21	-139
$A2 + D4$ 5 24 -50 -4 19 33 -34 50 7 -14 -39 28 -36 -22 -50 -21 -20 -7 48 29 -10 18 -33 -31 5	-70

A2-D4+C1+C2+C3 12 31 -39 34 -12 -46 0 -29 -17 -34 -4 -45 -38 12 5 38 17 -3 17 19 27 40 36 22 -7	-96
D4*C1*C2-C3 -26 38 -30 42 35 26 -22 45 38 -6 41 49 6 -38 -42 -37 -4 -20 -4 31 47 -40 25 44 -6	11900

Пример решения

```
f = input()
sf = f
mass = []
for i in range(5):
    mass.append(list(map(int, input().split())))
f = f.replace("A", "0").replace("B", "1").replace("C", "2").replace("D", "3").replace("E", "4")
s = ""
c = ""
for let in f:
    if let != "*" and let != "+" and let != "-":
        c += let
    else:
        s += str(mass[int(c[1])][int(c[0])])
        c = ""
        s += let
s += str(mass[int(c[1])][int(c[0])])

print(eval(s))
```

Задача 2 (12 баллов) id 5138

Условие

ИТ - компания "Пунк" решила создать программу, которая помогала бы пользователям мгновенно трансформировать текст, ошибочно написанный не на той раскладке языка. Чаще всего такая проблема возникает с написанием русских слов на английской раскладке клавиатуры.

На вход программы поступает строка, состоящая из английских символов нижнего регистра от а до z, а также символы "[", "]", ",", ":", ";", ".", " ". Длина строки $5 \leq X \leq 300$.

Считается, что знаки препинания были написаны корректно. В предложениях не используются знаки препинания “.” и “,”.

На выходе программа должна выдать строку с трансформированными символами с английской раскладки на русскую. Однако из-за особенностей компьютера он не может корректно обработать русские символы, поэтому программисты временно выводят номера каждого символа разделяя их символом “\”. ID символа выводится согласно таблице ASCII.

Например: “[jxtim .ke” , должно преобразоваться в “хочешь юлу”, а на выходе получится “u0445\u043e\u0447\u0435\u0448\u044c \u044e\u043b\u0443”. Символ пробела остаётся пробелом и не переводится в кодировку.

Входные Выходные данные:
данные:

[jxtim .ke	u0445\u043e\u0447\u0435\u0448\u044c \u044e\u043b\u0443
d yfitq ;bpyb rf;lsq lty m ghjbc[jlbn xnj-nj yjdjt b bynthtcyjt	u0432 \u043d\u0430\u0448\u0435\u0439 \u0436\u0438\u0437\u043d\u0438 \u0430\u0430\u0436\u0434\u0439 \u0434\u0430\u044c \u043f\u0440\u0438\u0441\u0442\u0430 \u0442\u0435\u043c\u0435 \u043d\u0435 \u0432\u0435\u0434\u0430\u0442\u044c \u0432 \u043a\u043e\u0434\u0438\u0440\u043e\u0432\u043a\u0443

Проверочные тесты

Входные данные	Ожидаемый результат
[jxtim .ke?	u0445\u043e\u0447\u0435\u0448\u044c \u044e\u043b\u0443
d yfitq ;bpyb rf;lsq lty m ghjbc[jlbn xnj-nj yjdjt b bynthtcyjt	u0432 \u043d\u0430\u0448\u0435\u0439 \u0436\u0438\u0437\u043d\u0438 \u0430\u0430\u0436\u0434\u0439 \u0434\u0430\u044c \u043f\u0440\u0438\u0441\u0442\u0430 \u0442\u0435\u043c\u0435 \u043d\u0435 \u0432\u0435\u0434\u0430\u0442\u044c \u0432 \u043a\u043e\u0434\u0438\u0440\u043e\u0432\u043a\u0443

gentitcndbz gjjvuf.n hfcibhbnm rheujpjh b epynfm vyuj yjdjuj	u043f\u0443\u0442\u0435\u0448\u0435\u0441\u0442\u0432\u0438\u044f \u043f\u043e\u043c\u043e\u0433\u0433\u0430\u044e\u0442 \u0440\u0430\u0441\u0441\u0448\u0438\u0440\u0438\u0442\u044c \u043a\u0440\u0443\u0433\u043e\u0437\u043e\u0440 \u0438 \u0443\u0437\u043d\u0430\u0442\u044c \u043c\u043d\u043e\u0433\u043e \u043d\u043e\u0432\u043e\u043e\u0433\u043e
xntybt rybu gjpdjktzn jreyenmcz d vbh afynfpbq b ghbrk.xtybq	u0447\u0442\u0435\u043d\u0438\u0435 \u043a\u043d\u0438\u0433 \u043f\u043e\u0437\u0437\u0432\u043e\u0432\u043e\u0432\u0435\u0432\u0435\u0432\u0435\u0442 \u043e\u043a\u0443\u043d\u0443\u043d\u0442\u0442\u044c\u0441\u044f \u0432\u0432 \u043c\u0438\u0442\u0442\u0440\u0442\u0442\u0442\u0442\u0442\u0442 \u0432\u0438 \u0444\u0430\u043d\u043d\u0442\u0430\u0437\u0438\u0438 \u0438 \u043f\u0440\u0430\u0442\u0430\u0432\u043e\u0442\u0435\u0442\u0435\u0442\u0435\u0442\u0435\u0442\u0435 9
enhtyyzz pfhzlrf gjvjuftn pfhzbncmz 'ythubtq yf dtcm lty	u0443\u0442\u0440\u0435\u043d\u043d\u044f\u044f \u0437\u0430\u0440\u0440\u044f\u0432\u0430\u0430 \u043f\u043e\u043c\u043e\u043e\u0433\u0433\u0430\u0435\u0442 \u0437\u0430\u0440\u0440\u044f\u0432\u0438\u0442\u044c\u0441\u044f \u044d\u043d\u0435\u0440\u0435\u0440\u0433\u0438\u0435\u0439 \u043d\u0430 \u0432\u0435\u0441\u0442\u0435\u0442\u0435\u0442\u0435\u0442\u0435\u0442\u0435
j,otybt c lhepmzvb b ,kbrpbvb k.lmnb cfvt wtyyt xnj e yfc tcnm	u043e\u0431\u0449\u0435\u043d\u0438\u0435 \u0435 \u0441 \u0432\u0440\u0443\u0443\u0437\u0442\u044c\u044f\u043c\u0438\u0438 \u0438 \u0431\u0432\u0438\u0438\u0437\u0430\u0432\u0438\u043c\u0438 \u0432\u0442\u0430\u0442\u0432\u0432\u043c\u0438 \u0441\u0430\u043c\u043e\u043e\u0435 \u0446\u0435\u043d\u043d\u0435\u0435 \u0442\u0442\u0442\u0442\u0442 \u043d\u0430\u0430\u0441 \u0435\u0442\u0442\u0442\u0442
exbnmcz ybrjulf yt gjplyj b rf;lsq vj;tn epyfnn xnj-nj yjdjt	u0443\u0447\u0438\u0442\u0442\u0442\u0442\u0441\u044f \u043d\u0438\u0432\u0430\u043e\u043e\u0433\u0432\u0434\u0430 \u043d\u0438 \u043f\u043e\u0437\u0437\u0432\u0430\u043d\u043e \u0438 \u043a\u0430\u0430\u0436\u0432\u0432\u0442\u0442\u0439 \u043c\u043e\u0432\u0435\u0435\u0442 \u0443\u0437\u0437\u043d\u0430\u0442\u0442\u0442\u0442 \u0442\u0442\u0442\u043e- \u0442\u043e \u043d\u0438\u0432\u0432\u043e\u0432\u0435\u0435
ghbhjlf yfi lju b vs ljk;ys pf,jnbnmcz j ytq b cj[hfyznm tt rhfcjne	u043f\u0440\u0438\u0440\u043e\u043e\u043e\u0432\u0430\u0430 \u043d\u0438 \u0432\u0432\u043e\u043c \u0438 \u0432\u0432 \u0432\u0432\u0432\u0436\u043d\u0442\u0442 \u0437\u0430\u0440\u0431\u0442\u0438\u0442\u0442\u0442\u0442\u0442\u0442 \u0441\u044f \u043e \u043d\u0438\u0435\u0439 \u0438 \u0441\u043e\u0442\u0440\u0442\u0430\u043d\u0435\u0442\u0442\u0442 \u0435\u0442\u0442\u0442 \u0442\u0442\u0442

Пример решения

```
sa="qwertyuiop[]asdfghjkl;'zxcvbnm,."
sr="йцукенгшщзхъфывапролджэячсмитьбю"
dictionary={}
for i in range(len(sa)):
    dictionary[sa[i]]=sr[i]
n=input()
fs=""
for i in n:
    if i in dictionary:
        fs+=dictionary[i]
    else:
        fs+=i
ans=ascii(fs)[2:-1]
if ans[-1] in '?!':
    ans=ans[:-1]
print(ans)
```

Задача 3 (16 баллов) id 5149

Условие

Алексей живёт в Москве (Moscow) и на новый год решил отправиться к своей бабушке в далёкий город Малокрибирск (Malokribirsk). До этого города никогда не летают самолёты, поэтому он решил добраться до него на поезде. Но прямые поезда ходят крайне редко, поэтому добираться придётся с пересадками. Алексей выписал все возможные маршруты поездов, которые могут ему пригодиться, включая информацию о том, откуда отправляется, куда прибывает, в какое время по Московскому времени он отправляется, через какое время прибывает. Алексею нужно некоторое время, чтобы успеть на следующий поезд при пересадке, на что требуется минимум 30 минут. Помогите Алексею определить, сможет ли он добраться до своей бабушки минимум чем за 92 часа, притом за самое минимальное время, которое возможно. Если добраться невозможно, вывести только цифру 0.

Входные данные:

В первой строке подаётся число N ($1 \leq N \leq 1000$) – количество билетов, далее на N -строках подаются все возможные варианты маршрутов в формате город отправления, город прибытия, время отправления по МСК (в формате час:минут), время в пути (в формате час:минут), разделённые вертикальной чертой. Считаем, что поезда отправляются **ЕЖЕДНЕВНО**.

Выходные данные:

Выведите на первой строке маршрут, который сможет проехать Алексей, на второй строке время в пути. Либо выведите только 0, если Алексей не сможет добраться до бабушки за 92 часа.

Ввод	Вывод
4 Moscow Saint Petersburg 10:15 8:15 Vladivostok Novosibirsk 12:00 35:50 Saint Petersburg Novosibirsk 19:15 42:10 Novosibirsk Malokribirsk 20:00 3:05	Moscow Saint Petersburg Novosibirsk Malokribirsk 60:50
4 Moscow Saint Petersburg 10:15 8:15 Vladivostok Novosibirsk 12:00 35:50 Saint Petersburg Novosibirsk 12:15 42:10 Novosibirsk Malokribirsk 20:00 3:05	Moscow Saint Petersburg Novosibirsk Malokribirsk 84:50

Пример №2

Алексей может отправиться из Москвы только в Санкт-Петербург. Едет поезд 8 часов 15 минут. Значит прибудет в 18:30. До ближайшего поезда остаётся 17 часов 45 минут. Но поезда ходят каждый день, таким образом он сможет отправить на поезде на следующий день. Далее на поезде проедет до Новосибирска, а оттуда до Малокрибирска. Получится маршрут длительностью 84 часа и 50 минут, что меньше 92 часов, значит нам подходит. И он самый минимальный.

Примечание:

Гарантируется, что циклов быть не может.
Посещать один и тот же город нельзя.

Проверочные тесты

Входные данные	Ожидаемый результат
4 Moscow Saint Petersburg 10:15 8:15 Vladivostok Novosibirsk 12:00 35:5 0 Saint Petersburg Novosibirsk 19:15 42:1 0 Novosibirsk Malokribirsk 20:00 3:0 5	Moscow Saint Petersburg Novosibirsk Malokribirsk 60:50
4 Moscow Saint Petersburg 10:15 8:15 Vladivostok Novosibirsk 12:00 35:5 0 Saint Petersburg Novosibirsk 12:15 42:1 0 Novosibirsk Malokribirsk 20:00 3:0 5	Moscow Saint Petersburg Novosibirsk Malokribirsk 84:50
11 Nalchik Vladikavkaz 14:58 03:57 Samara Rybinsk 04:59 09:55 Nadym Ryazan 17:32 09:12 Murom Saransk 12:08 04:33 Saratov Salekhard 08:49 01:46 Salekhard Murmansk 12:43 01:41 Rybinsk Saransk 14:17 04:33 Vladikavkaz Saratov 00:26 06:48 Moscow Vladikavkaz 16:08 08:09 Salekhard Malokribirsk 01:09 04:3 7 Samara Moscow 23:07 02:45	Moscow Vladikavkaz Saratov Salekhard Malokribirsk 61:38

14 Samara Naberezhnye Chelny 16:34 06:40 Saratov Severodvinsk 15:15 04:14 Malokribirsk Vologda 13:17 04:58 Malokribirsk Vladikavkaz 16:13 00:44 Murmansk Nadym 08:34 05:46 Rybinsk Saratov 23:04 04:56 Severodvinsk Malokribirsk 10:12 07:08 Severodvinsk Malokribirsk 06:24 05:58 Moscow Ryazan 06:57 06:34 Ryazan Saratov 09:28 07:07 Malokribirsk Salekhard 05:06 07:01 Murmansk Malokribirsk 09:24 08:38 Nalchik Severodvinsk 17:16 05:15 Salekhard Moscow 00:05 02:04	Moscow Ryazan Saratov Severodvinsk Malokribirsk 77:25
12 Naberezhnye Chelny Samara 14:12 09:28 Nalchik Malokribirsk 20:13 08:46 Samara Murmansk 12:07 05:05 Vladikavkaz Rybinsk 16:02 05:55 Salekhard Saint Petersburg 20:37 08:39 Murmansk Nalchik 08:07 08:56 Moscow Samara 14:13 06:18 Saratov Saint Petersburg 22:56 04:28 Nalchik Malokribirsk 16:01 01:59 Saransk Vladimir 07:01 05:00 Vladikavkaz Salekhard 04:08 09:36 Vladikavkaz Saratov 17:55 01:38	Moscow Samara Murmansk Nalchik Malokribirsk 62:46

14 Volgograd Salekhard 11:44 06:31 Murom Vorkuta 23:55 03:39 Moscow Volgograd 22:08 07:29 Rybinsk Vologda 07:01 09:50 Volgograd Naberezhnye Chelny 18:54 03:46 Vorkuta Malokribirsk 08:08 08:55 Malokribirsk Rybinsk 10:21 03:39 Murmansk Saransk 05:11 02:42 Salekhard Vladimir 06:43 07:39 Vladimir Vorkuta 14:31 09:56 Rybinsk Vologda 23:19 06:10 Saransk Malokribirsk 13:36 07:28 Saint Petersburg Murmansk 22:54 06:41 Volgograd Naberezhnye Chelny 06:28 07:56	Moscow Volgograd Salekhard Vladimir Vorkuta Malo kribirsk 90:55
15 Nadym Volgograd 05:51 06:53 Nalchik Murom 01:56 01:32 Vorkuta Nadym 06:06 03:31 Samara Murom 10:08 08:37 Saratov Saransk 04:32 03:47 Murmansk Naberezhnye Chelny 10:58 04:10 Salekhard Vorkuta 04:30 09:30 Murmansk Vorkuta 14:29 01:09 Nadym Malokribirsk 10:33 09:23 Murom Saint Petersburg 12:07 08:38 Nadym Malokribirsk 18:33 08:03 Salekhard Malokribirsk 03:25 07:2 2 Severodvinsk Saratov 01:44 01:35 Moscow Salekhard 14:41 07:01 Murmansk Vologda 22:09 09:19	Moscow Salekhard Malokribirsk 20:06

15 Naberezhnye Chelny Vladikavkaz 21:23 09:11 Vladimir Nadym 22:49 05:32 Samara Saratov 17:27 08:25 Vladikavkaz Nadym 22:30 05:10 Nadym Vorkuta 01:53 07:05 Saransk Salekhard 06:41 07:18 Murom Vologda 06:38 00:53 Moscow Vladimir 00:52 04:54 Murmansk Volgograd 04:32 09:10 Vladimir Murom 01:34 00:57 Malokribirsk Moscow 01:14 03:48 Saint Petersburg Malokribirsk 21:25 00:4 6 Malokribirsk Volgograd 11:10 01:3 7 Murom Saint Petersburg 03:50 01:36 Vladikavkaz Vladimir 23:03 03:13	Moscow Vladimir Murom Saint Petersburg Malokribirsk 45:19
7 Salekhard Saransk 22:43 09:46 Naberezhnye Chelny Salekhard 08:11 02:07 Murom Severodvinsk 03:10 00:36 Salekhard Malokribirsk 05:53 02:3 6 Moscow Naberezhnye Chelny 10:49 04:58 Ryazan Saratov 11:59 01:22 Saransk Malokribirsk 14:03 03:49	Moscow Naberezhnye Chelny Salekhard Malokribirsk 45:40

15 Murmansk Volgograd 15:12 04:13 Nalchik Murmansk 09:32 03:47 Murom Malokribirsk 22:22 01:36 Nadym Murom 05:28 04:54 Salekhard Vorkuta 11:18 03:20 Rybinsk Murom 22:02 01:22 Moscow Vladikavkaz 12:55 05:24 Vorkuta Saint Petersburg 05:27 02:47 Saratov Vorkuta 06:12 03:15 Murmansk Malokribirsk 11:41 02:2 2 Vologda Murom 23:15 07:10 Vladikavkaz Samara 02:37 08:42 Samara Murmansk 04:57 02:21 Moscow Nalchik 19:51 04:52 Nalchik Vologda 00:02 04:17	Moscow Nalchik Murmansk Malokribirsk 42:12
9 Severodvinsk Vladikavkaz 20:31 0 1:52 Samara Vorkuta 02:57 09:06 Vologda Severodvinsk 03:39 02:25 Vladimir Malokribirsk 16:31 04:12 Vorkuta Vladimir 14:05 03:49 Saint Petersburg Vologda 15:24 09:48 Moscow Samara 14:12 08:37 Murmansk Volgograd 16:57 08:48 Murmansk Samara 18:37 01:41	Moscow Samara Vorkuta Vladimir Malokribirsk 54:31
14 Vladikavkaz Malokribirsk 16:10 04: 50 Volgograd Salekhard 16:38 08:51 Rybinsk Vologda 06:21 07:07 Murmansk Saratov 04:48 06:54 Ryazan Saint Petersburg 17:19 06:28 Salekhard Saint Petersburg 08:22 07:45 Moscow Vologda 00:32 04:48 Nalchik Naberezhnye Chelny 13:57 09:47	Moscow Vologda Ryazan Saint Petersburg Vladikavkaz Malokribirsk 68:28

Volgograd Vologda 09:56 08:01 Saint Petersburg Vladikavkaz 01:51 03:47 Vologda Ryazan 18:49 09:50 Vorkuta Saransk 13:35 02:42 Malokribirsk Vologda 01:15 03:15 Nalchik Vologda 07:06 06:07	
15 Naberezhnye Chelny Malokribirsk 16:10 02:48 Murom Salekhard 14:31 03:43 Vladimir Ryazan 23:57 03:51 Vologda Salekhard 02:48 09:06 Murom Salekhard 06:34 08:50 Murmansk Severodvinsk 19:35 09:25 Murmansk Naberezhnye Chelny 00:05 03:37 Moscow Salekhard 17:23 06:50 Moscow Volgograd 07:40 05:54 Volgograd Vorkuta 17:52 05:53 Vladikavkaz Malokribirsk 03:37 08:55 Vorkuta Murmansk 18:36 08:53 Murom Naberezhnye Chelny 17:36 03:29 Rybinsk Nalchik 22:56 01:52 Saratov Vladikavkaz 09:11 04:12	Moscow Volgograd Vorkuta Murmansk Naberezhnye Chelny Malokribirsk 83:18

15 Rybinsk Saransk 18:34 04:54 Saransk Malokribirsk 12:51 01:22 Vorkuta Samara 03:47 02:49 Vorkuta Nalchik 03:44 02:58 Murmansk Vorkuta 16:37 08:18 Severodvinsk Salekhard 17:03 06:10 Saransk Naberezhnye Chelny 06:16 02:57 Rybinsk Malokribirsk 04:21 05:39 Murmansk Malokribirsk 10:14 02:28 Saint Petersburg Saratov 21:11 03:38 Naberezhnye Chelny Severodvinsk 15:00 06:16 Vorkuta Moscow 17:28 01:53 Samara Rybinsk 22:47 05:53 Nadym Samara 10:37 00:35 Moscow Vorkuta 03:56 05:39	Moscow Vorkuta Samara Rybinsk Malokribirsk 78:04
11 Nalchik Saransk 17:43 03:40 Severodvinsk Saint Petersburg 18:23 00:41 Saransk Murmansk 04:47 05:21 Severodvinsk Salekhard 16:59 08:39 Vladikavkaz Rybinsk 11:53 07:00 Severodvinsk Naberezhnye Chelny 02:59 09:43 Rybinsk Murmansk 13:44 09:55 Moscow Vladikavkaz 02:06 01:28 Saransk Vladikavkaz 09:07 04:50 Saint Petersburg Malokribirsk 01:56 02:18 Murmansk Malokribirsk 20:33 04:40	Moscow Vladikavkaz Rybinsk Murmansk Malokribirsk 71:07

14 Moscow Murom 01:51 07:22 Vladimir Malokribirsk 10:12 05:35 Severodvinsk Murmansk 08:02 06:29 Vladikavkaz Naberezhnye Chelny 07:54 01:28 Malokribirsk Vorkuta 22:24 04:36 Vladikavkaz Vladimir 20:34 07:14 Nadym Vladikavkaz 21:58 06:31 Murom Nadym 18:29 06:26 Nalchik Saratov 16:06 04:20 Murmansk Ryazan 03:24 05:22 Vologda Nadym 14:29 07:53 Ryazan Samara 15:23 01:10 Samara Malokribirsk 18:31 02:28 Volgograd Saratov 08:03 09:11	Moscow Murom Nadym Vladikavkaz Vladimir Malokribirsk 85:56
13 Naberezhnye Chelny Saransk 19:08 09:23 Nadym Malokribirsk 18:17 04:24 Volgograd Nadym 13:56 05:57 Samara Ryazan 23:43 06:22 Nalchik Malokribirsk 19:58 05:24 Moscow Murom 22:40 09:10 Ryazan Vologda 08:53 05:36 Vologda Murmansk 20:28 06:02 Nalchik Saransk 15:22 05:27 Murom Volgograd 19:30 02:06 Saratov Saint Petersburg 22:26 09:11 Saratov Moscow 18:43 01:52 Rybinsk Volgograd 14:46 03:35	Moscow Murom Volgograd Nadym Malokribirsk 72:01

14 Vladikavkaz Nadym 10:14 03:36 Vorkuta Vladimir 21:15 03:49 Salekhard Vorkuta 23:59 09:26 Murom Moscow 02:47 08:29 Moscow Vologda 00:18 05:14 Nalchik Nadym 18:38 07:26 Moscow Saratov 01:00 06:21 Murmansk Murom 20:52 03:42 Saransk Saratov 21:40 07:15 Saratov Malokribirsk 23:35 05:30 Vologda Saransk 23:31 06:10 Vologda Saint Petersburg 02:25 08:58 Nalchik Vorkuta 08:10 02:42 Moscow Ryazan 08:04 01:44	Moscow Saratov Malokribirsk 28:05
14 Vladimir Salekhard 19:12 09:13 Volgograd Naberezhnye Chelny 13:53 02:34 Nadym Nalchik 08:35 06:01 Naberezhnye Chelny Malokribirsk 01:57 08:13 Moscow Nadym 22:01 00:51 Vladimir Nalchik 13:49 09:07 Vladikavkaz Murom 23:30 01:27 Naberezhnye Chelny Vladikavkaz 12:50 02:14 Severodvinsk Volgograd 19:06 00: 30 Vologda Murom 12:33 01:12 Malokribirsk Severodvinsk 02:14 0 0:49 Volgograd Saransk 20:09 08:03 Salekhard Samara 11:32 09:54 Nalchik Naberezhnye Chelny 09:10 06:09	Moscow Nadym Nalchik Naberezhnye Chelny Malokribirsk 60:09

14 Murom Moscow 10:12 01:05 Moscow Saint Petersburg 11:45 07:07 Vologda Murom 02:40 05:37 Vorkuta Malokribirsk 10:26 00:41 Vologda Vladikavkaz 15:46 08:46 Salekhard Vorkuta 23:15 13:40 Samara Vorkuta 10:16 04:18 Vologda Vladikavkaz 02:28 02:01 Saint Petersburg Salekhard 18:05 05:33 Salekhard Nalchik 14:41 02:13 Saratov Saint Petersburg 01:03 08:34 Malokribirsk Vladimir 21:47 04:01 Vladikavkaz Murom 02:50 07:46 Murmansk Volgograd 12:12 03:23	0
---	---

Пример решения

```
def train(city, route, cur_time, sum_time):
    global routs
    global out_routes
    if city == 'Malokribirsk':
        out_routes.append((route, sum_time))
        return None
    elif city not in routs:
        return None
    else:
        for t in routs[city]:
            if (t[1] - cur_time) % 1440 >= 30:
                train(t[0], route.copy() + [t[0]], (t[1] + t[2]) % 1440, sum_time + (t[1] - cur_time) %
1440 + t[2])
            else:
                train(t[0], route.copy() + [t[0]], (t[1] + t[2]) % 1440, sum_time + 1440 + t[1] -
cur_time + t[2])

n = int(input())
out_routes = []
count = 0
routs = {}
for _ in range(n):
    rout = input().split('|')
    if rout[0] in routs:
```

```

    h, m = [int(i) for i in rout[2].split(':')]
    h1, m1 = [int(i) for i in rout[3].split(':')]
    routs[rout[0]] = routs[rout[0]] + [[rout[1], h * 60 + m, h1 * 60 + m1]]
else:
    h, m = [int(i) for i in rout[2].split(":")]
    h1, m1 = [int(i) for i in rout[3].split(':')]
    routs[rout[0]] = [[rout[1], h * 60 + m, h1 * 60 + m1]]

if 'Moscow' in routs:
    for t in routs['Moscow']:
        train(t[0], ['Moscow', t[0]], (t[1] + t[2]) % 1440, t[2])

if out_routes:
    out_routes = sorted(out_routes, key=lambda x: x[1])

    if out_routes[0][1] <= 5520:
        a = out_routes[0][0]
        print('|'.join(a))
        time = out_routes[0][1]
        h, m = time // 60, time % 60

        if h == 0:
            if m < 10:
                print(f'0:0{m}')
            else:
                print(f'0:{m}')
        else:
            if m < 10:
                print(f'{h}:0{m}')
            else:
                print(f'{h}:{m}')
        else:
            print(0)
    else:
        print(0)

```

Задача 4 (20 баллов) id 5140

Условие

Пете на день рождения подарили Робота, однако робот оказался таким, что его нужно обучать. Петя решил обучить робота находить на клеточном поле самый большой параллелограмм и выводить его площадь. Для обучения Петя решил написать программу, которая будет получать на вход клеточное поле в виде матрицы из нулей и единиц, где нули - это свободные клетки, а единицы - закрашенные. Петя очень долго возился с кодом и понял, что у него ничего не получается, поэтому он обратился к Вам за помощью. Помогите ему написать программу, которая по входной матрице будет

находить самый большой параллелограмм из пустых клеток, а также выводить его площадь. Петя гарантирует, что в его матрице такой параллелограмм существует 100%. Но важно также учесть, что параллелограмм должен быть **ОБЯЗАТЕЛЬНО** ограничен со всех сторон единицами и не касаться границ клеточного поля, иначе это будет уже совсем другая фигура.

Входные данные:

На первой строке подаются размеры матрицы N – количество строк ($4 \leq N \leq 1000$) и M – кол-во столбцов ($4 \leq M \leq 1000$). Далее на N строках через пробел введено по M целых чисел матрицы – числа клеточного поля (0 или 1).

Выходные данные:

На первой строке необходимо вывести максимальную площадь параллелограмма, которая есть на данном клеточном поле (целое число без запятой или точки!). На второй строке требуется вывести координаты левой верхней клетки начала параллелограмма через пробел в формате (номер_строки номер_столбца (нумерация с ЕДИНИЦЫ)).

Ввод	Вывод
<pre> 4 10 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 </pre>	<pre> 6 2 2 </pre>
<pre> 7 15 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 0 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 </pre>	<pre> 12 2 8 </pre>

Примечание:

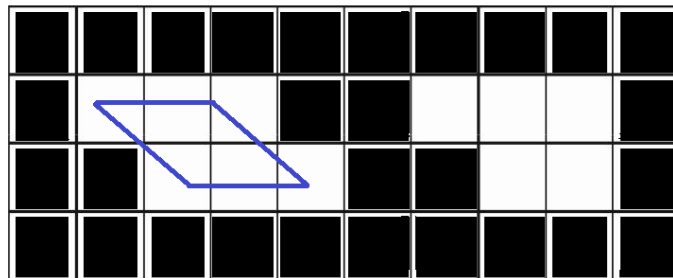
- гарантируется, что параллелограмм всегда будет расположен также, как в приведённых примерах, никак иначе (без поворотов и отражений)!

Пример №1

```

1 1 1 1 1 1 1 1 1 1
1 0 0 0 1 1 0 0 0 1
1 1 0 0 0 1 1 0 0 1
1 1 1 1 1 1 1 1 1 1

```

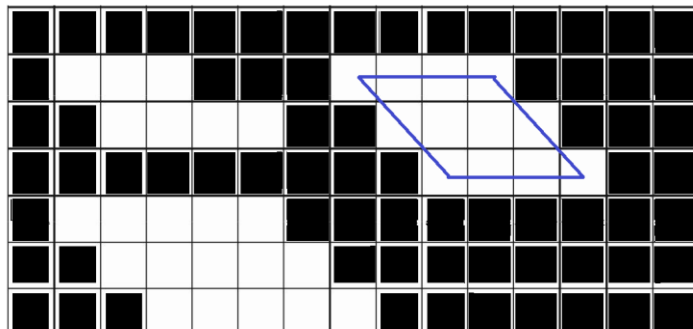


Пример №2

```

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 0 1 1 1 0 0 0 0 1 1 1 1 1
1 1 0 0 0 0 1 1 0 0 0 0 1 1 1 1
1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1
1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1
1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1
1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1

```



Параллелограмм, ограниченный единицами здесь только один, поэтому его площадь и максимальна.

Проверочные тесты

Входные данные	Ожидаемый результат
4 10 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1	6 2 2

[illegible]

29 2

111000001111111111000000111111	
111111111111111111111111111111	
111111111111111111111111111111	
100011100000111100011100000111	
110000110000111110000110000111	
11111111100001111111111000011	
100000111111111110000001111111	
110000011111111111000000111111	
111000001111111111000000011111	
111111111111111111111111111111	
111111111111111111111111111111	
100011100000111100011100000111	
110000110000011110000110000111	
1111111110000011111111111000011	
100000111100000100000011111111	
110000011110000110000001111111	
111000001111111111100000011111	
111111111111111111111111111111	
111111111111111111111111111111	
100011100000111100011100000111	
110000110000111110000110000111	
111111111000011111111111000011	
100000111111111110000001111111	
110000011111111110000001111111	
111000001111111111000000011111	
111111111111111111111111111111	
111111111111111111111111111111	
100011100000111100011100000111	
110000110000011110000110000111	
111111111000011111111111000011	
100000111111111110000001111111	
110000011111111110000001111111	
111000001111111111000000011111	
111111111111111111111111111111	
111111111111111111111111111111	
100011100000111100011100000111	
110000110000011110000110000111	
111111111000001111111111000011	
100000111100000100000011111111	
110000011110000110000001111111	
111000001111111111100000011111	
111111111111111111111111111111	
111111111111111111111111111111	
100011100000111100011100000111	
110000110000011110000110000111	
111111111000001111111111000011	
100000111100000100000011111111	
110000011110000110000001111111	
111000001111111111100000011111	
111111111111111111111111111111	

Пример решения

```
N, M= map(int, input().split())
matrix=[]
for i in range(N):
    matrix.append(list(map(int, input().split())))

k=0
max_s=0
ind_i=0
ind_j=0
for i in range(1, N-1):
    for j in range(1, M-1):
        if matrix[i][j]==0:
            k+=1
        else:
            if k==0:
                continue
            else:
                q_s=k
                c = i
                d = j - k
                while True:
                    c+=1
                    d+=1
                    if all(c<N-1 and z<M-1 and matrix[c][z]==0 for z in range(d, d+k)) and
matrix[c][d-1]==1 and matrix[c][d+k]==1:
                        q_s+=k
                        continue
                    else:
                        if all(c<=N-1 and z<=M-1 and matrix[c][z]==1 for z in range(d, d + k)):
                            break
                        q_s=0
                        break

                if q_s>max_s:
                    max_s=q_s
                    ind_i=i+1
                    ind_j=j-k+1
                k=0
print(max_s)
print(ind_i, ind_j)
```

Задача 5 (20 баллов) id 5144

Условие

Иван Петрович проснулся в один день от звонка адвоката. Он никак не ожидал, что темой звонка будет то, что в наследство ему достался загородный участок от дедушки. Иван Петрович помнил, как в детстве он играл там постоянно со своими друзьями, своей любимой собакой, но также помнил и о своей мечте - построить треугольный сквер. После новости о наследстве, Иван Петрович запросил план участка, с указанием координат каждой посаженной берёзы, чтобы понять, можно ли осуществить задуманное, или придётся вырубить все берёзы и посадить их заново. Для этого Иван Петрович решил заказать у Вас расчёт плана о вырубке тех деревьев, которые нужно вырубить.

План вырубки следующий – нужно построить сквер так, чтобы он представлял из себя треугольник в вершинах которого находится по одной берёзе, а внутри него содержалось максимальное количество других берёз, а по каждой стороне вне треугольника было одинаковое количество оставшихся берёз. Однако если осуществить задуманное невозможно, то необходимо сообщить Ивану Петровичу об этом.

Входные данные

На первой строке вводится число N – количество деревьев на участке ($4 \leq N \leq 20$). Далее на N строках указываются пары координат деревьев на участке, в виде вещественных чисел (точность координат до 8 знаков после запятой).

Выходные данные

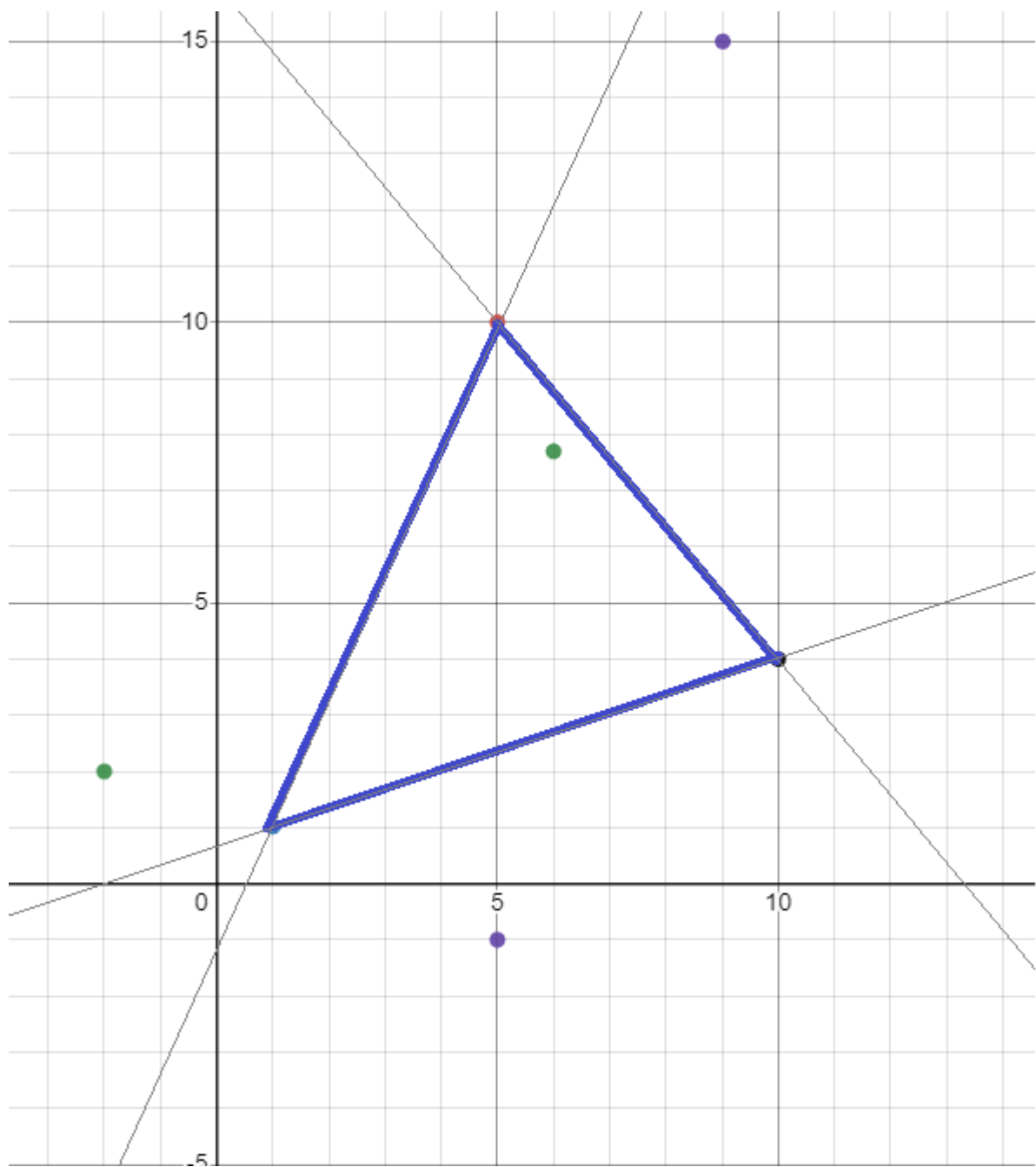
На трёх строках вывести координаты деревьев (округлив до двух знаков; если значение целое, то выводить ВСЕГДА два знака (например 5.00, либо 1.70)), которые образуют искомые координаты треугольного сквера (его вершины), начиная от вершины с максимальным Y и минимальным X , а далее против часовой стрелке. Если же задачу решить невозможно, то необходимо вывести на первой строке 0, а далее вывести в отсортированном порядке по двум осям координаты всех берёз для удобства вырубки (с двумя знаками после запятой; если значения целые, то ВСЕГДА выводить два знака (например, 5.00, 1.70)).

Ввод	Вывод
------	-------

7	5.00 10.00
6.00 7.70	1.00 1.02
9.00 15.00	10.00 4.00
10.00 4.00	
5.00 10.00	
1.00 1.0175	
-2.00 2.00	
5.00 -1.00	
5	0
31.914 0.6194	28.09 0.62
31.1829 -1.6306	28.82 -1.63
30.00 0.62	30.00 0.62
28.8171 -1.6306	31.18 -1.63
28.086 0.6194	31.91 0.62

Пример 1.

В данном примере самый максимальный треугольник можно построить как на рисунке ниже.



Проверочные тесты

Входные данные	Ожидаемый результат
----------------	---------------------

7 6.00 7.70 9.00 15.00 10.00 4.00 5.00 10.00 1.00 1.0175 -2.00 2.00 5.00 -1.00	5.00 10.00 1.00 1.02 10.00 4.00
5 31.914 0.6194 31.1829 -1.6306 30.00 0.62 28.8171 -1.6306 28.086 0.6194	0 28.09 0.62 28.82 -1.63 30.00 0.62 31.18 -1.63 31.91 0.62
7 489.0 110.5 235.0 378.0 306.0 56.0 402.0 476.5 244.5 383.5 -371.0 276.0 280.0 -455.5	402.00 476.50 235.00 378.00 280.00 -455.50
12 498.5 203.0 -469.0 417.5 -352.0 -193.0 50.0 172.5 -318.5 399.0 344.0 265.0 322.0 -24.0 97.5 278.5 310.0 -108.0 42.0 233.5 -301.5 -405.5 -245.5 -267.0	-469.00 417.50 97.50 278.50 310.00 -108.00

11 31.0 -18.0 122.5 498.5 263.5 54.0 127.0 475.5 -13.0 273.5 95.0 380.5 333.5 371.0 408.5 332.0 -348.0 145.5 -219.5 464.0 383.5 -103.0	127.00 475.50 -13.00 273.50 383.50 -103.00
18 321.0 -14.5 -85.0 25.5 -498.5 -460.0 354.5 -465.0 -391.5 433.0 -474.0 250.0 -69.0 -474.0 -127.0 124.5 -225.5 -405.0 61.5 -131.0 146.5 87.0 79.0 416.0 -411.0 449.5 -1.5 461.0 -107.0 107.0 -76.5 -232.5 81.5 484.0 -222.5 131.0	-411.00 449.50 -76.50 -232.50 146.50 87.00

17 463.5 -395.0 80.0 -485.5 -107.5 -454.5 -25.5 -427.0 -271.0 -329.5 213.0 -180.5 -244.0 158.5 233.0 428.5 452.0 -35.5 -126.5 95.5 211.0 497.5 491.0 -315.5 16.5 460.5 249.5 -115.0 410.0 234.0 151.0 9.0 -263.5 360.0	-244.00 158.50 213.00 -180.50 452.00 -35.50
14 -463.5 -281.5 -474.5 126.5 -40.5 -200.5 -176.0 131.0 -40.5 -422.5 220.5 387.5 -98.0 -141.5 262.0 291.5 -432.5 431.0 -274.0 304.5 64.5 75.5 42.0 -177.5 -136.0 -463.5 -73.0 -92.5	220.50 387.50 -474.50 126.50 -40.50 -422.50

20 -271.5 -493.0 278.0 300.0 -396.0 244.5 155.5 -34.5 -413.5 -339.5 415.5 -93.0 91.0 335.0 22.0 256.5 -373.5 -464.5 375.0 490.0 462.0 -48.0 142.0 475.5 267.0 316.0 396.5 473.0 374.5 -342.0 -19.5 -344.0 99.5 470.5 -289.0 467.0 -281.0 127.5 -432.5 -155.0	396.50 473.00 -396.00 244.50 -19.50 -344.00
7 -106.0 98.5 479.5 -466.0 317.0 30.0 -424.0 -373.5 -200.0 318.0 369.0 81.5 390.5 47.0	-106.00 98.50 369.00 81.50 479.50 -466.00
7 -282.5 229.5 270.5 228.5 -179.5 -168.5 -290.0 308.5 241.5 155.5 -247.5 281.5 19.5 -74.0	-290.00 308.50 -179.50 -168.50 241.50 155.50

18 160.5 489.0 -88.5 32.0 330.5 -226.0 446.5 163.0 430.5 -435.0 52.0 -121.5 -456.0 121.5 488.5 25.0 104.0 -359.0 113.0 -46.0 130.0 69.5 447.5 106.0 -170.5 -23.0 -114.5 252.5 -114.5 -247.5 127.0 -413.0 -436.0 36.5 406.5 -209.5	160.50 489.00 104.00 -359.00 113.00 -46.00
16 490.5 -67.5 -194.0 -426.5 500.0 -454.5 383.5 -44.5 414.5 364.5 -404.0 -9.5 30.0 -63.5 2.0 137.0 -252.5 265.5 -170.0 -0.5 -350.0 274.0 -173.5 280.0 339.5 -421.0 185.0 -497.0 -124.5 -239.0 -16.5 -324.0	2.00 137.00 -404.00 -9.50 339.50 -421.00

15 422.0 11.0 -107.0 -275.5 249.0 298.0 -321.5 -225.0 156.0 436.0 326.5 457.5 176.0 -50.5 -289.5 -235.0 207.0 336.5 -232.0 318.0 -221.5 -346.5 136.5 417.0 259.0 440.0 0.5 -319.0 67.5 222.5	156.00 436.00 -289.50 -235.00 422.00 11.00
15 -349.5 -352.0 -11.5 479.5 267.5 -198.0 -25.5 -10.5 298.5 254.0 279.5 -75.0 -408.0 -171.0 295.5 237.5 210.5 416.0 49.5 -220.5 400.5 -106.5 336.5 367.5 379.5 7.0 410.0 -404.5 221.5 -176.5	210.50 416.00 49.50 -220.50 400.50 -106.50
9 -271.0 291.5 262.5 -373.5 140.5 -118.0 -136.0 -189.0 52.0 252.5 -160.0 303.5 398.5 321.0 -88.5 411.5 277.0 162.0	-160.00 303.50 52.00 252.50 262.50 -373.50

20 -152.5 211.5 357.0 28.5 20.0 -355.0 -6.5 178.5 391.0 351.5 449.5 -102.0 405.0 299.5 394.5 -295.5 -452.0 -298.0 -143.5 321.5 -40.5 109.5 -49.0 204.5 -64.5 -232.5 -70.0 292.0 144.5 414.0 -123.5 -113.5 10.0 -56.0 -384.0 -58.0 64.5 192.5 255.0 474.5	391.00 351.50 -152.50 211.50 20.00 -355.00
7 429.0 -47.0 -5.0 -454.5 462.0 -193.5 -337.5 409.0 447.5 -179.0 43.5 -481.0 -26.0 -56.5	-337.50 409.00 43.50 -481.00 447.50 -179.00
16 405.5 -236.5 -168.5 -407.0 449.5 -159.5 -104.0 61.0 319.0 -498.0 16.0 -26.5 -238.5 -277.0 -168.5 -442.5 -339.5 114.0 467.5 -451.5 -280.0 -424.0 -494.5 5.0 466.5 -334.0	16.00 -26.50 -238.50 -277.00 319.00 -498.00

-34.0 -407.5 60.5 -207.0 -411.0 -12.0	
8 -251.5 205.0 6.5 176.5 -218.5 119.0 416.5 391.0 -156.0 425.5 -127.0 -89.5 163.0 40.5 -367.0 193.5	0 -367.00 193.50 -251.50 205.00 -218.50 119.00 -156.00 425.50 -127.00 -89.50 6.50 176.50 163.00 40.50 416.50 391.00

Пример решения

```

import math
def getLine(p1, p2):
    if p2[0]-p1[0]==0:
        return [math.inf, p1[0]]
    k=(p2[1]-p1[1])/(p2[0]-p1[0])
    b=p1[1]-k*p1[0]

    return [k, b]
def getArea(a, b, c):
    return abs((a[0]-c[0])*(b[1]-c[1]) - (b[0]-c[0])*(a[1]-c[1]))

def inTriangle(p, a, b, c):
    return (getArea(a, b, c) >= getArea(p, b, c) + getArea(a, p, c) + getArea(a, b, p))

def sign(p1, p2, p3):
    return (p1[0]-p3[0])*(p2[1]-p3[1])-(p2[0]-p3[0])*(p1[1]-p3[1])

def PointInTriangle (pt, v1, v2, v3):
    d1 = sign(pt, v1, v2)

```

```
d2 = sign(pt, v2, v3)
```

```
d3 = sign(pt, v3, v1)
```

```
has_neg = (d1 < 0) or (d2 < 0) or (d3 < 0)
```

```
has_pos = (d1 > 0) or (d2 > 0) or (d3 > 0)
```

```
return not(has_neg and has_pos)
```

```
n = int(input())
```

```
p = []
```

```
for i in range(n):
```

```
    x, y = map(float, input().split())
```

```
    p.append((x, y))
```

```
maxCountIn = 0
```

```
ans = []
```

```
for i in range(n-2):
```

```
    for j in range(i+1, n-1):
```

```
        for k in range(j+1, n):
```

```
            a = p[i]
```

```
            b = p[j]
```

```
            c = p[k]
```

```
            lines = [getLine(a, b), getLine(a, c), getLine(b, c)]
```

```
            lines.sort()
```

```
            lines.reverse()
```

```
            p2 = p.copy()
```

```
            p2.remove(a)
```

```
            p2.remove(b)
```

```
            p2.remove(c)
```

```
            if not(b[0] - a[0]) * (c[1] - a[1]) - (c[0] - a[0]) * (b[1] - a[1]):
```

```
                continue
```

```
            countIn = 0
```

```
            p3 = p2.copy()
```

```
            for x in p3:
```

```
                if PointInTriangle(x, a, b, c):
```

```
                    countIn += 1
```

```
                    p2.remove(x)
```

```
            if countIn > maxCountIn:
```

```
                k1 = 0
```

```
                k2 = 0
```

```
                k3 = 0
```

```
                centerX = (a[0] + b[0] + c[0]) / 3
```

```
                centerY = (a[1] + b[1] + c[1]) / 3
```

```
                cP = (centerX, centerY)
```

```

for x in p2:
    d1 = (x[0] - a[0])*(b[1] - a[1]) - (x[1] - a[1])*(b[0] - a[0])
    d2 = (cP[0] - a[0]) * (b[1] - a[1]) - (cP[1] - a[1]) * (b[0] - a[0])

    if d1 * d2 <= 0:
        k1 += 1

    d1 = (x[0] - a[0]) * (c[1] - a[1]) - (x[1] - a[1]) * (c[0] - a[0])
    d2 = (cP[0] - a[0]) * (c[1] - a[1]) - (cP[1] - a[1]) * (c[0] - a[0])

    if d1 * d2 <= 0:
        k2 += 1

    d1 = (x[0] - c[0]) * (b[1] - c[1]) - (x[1] - c[1]) * (b[0] - c[0])
    d2 = (cP[0] - c[0]) * (b[1] - c[1]) - (cP[1] - c[1]) * (b[0] - c[0])

    if d1 * d2 <= 0:
        k3 += 1
    if k1 == k2 and k2 == k3:
        maxCountIn = countIn
        ans = [a, b, c]

ans.sort()
if len(ans)==0:
    print(0)
    p.sort()
    for el in p:
        print("%.2lf" % el[0], "%.2lf" % el[1])
else:
    ans.sort(key=lambda x:x[1], reverse=True)

    print("%.2lf %.2lf" % ans[0])
    if ans[1][0] < ans[2][0]:
        print("%.2lf %.2lf" % ans[1])
        print("%.2lf %.2lf" % ans[2])
    elif ans[1][0] == ans[2][0] and ans[1][1] > ans[2][1]:
        print("%.2lf %.2lf" % ans[1])
        print("%.2lf %.2lf" % ans[2])
    else:
        print("%.2lf %.2lf" % ans[2])
        print("%.2lf %.2lf" % ans[1])

```

Задача 6 (24 баллов) id 5147

Условие

Пикелёв – маленький городок, с малым количеством улочек и совсем без дорожек для бега. Наташа который день хочет начать бегать по утрам, но каждый раз её останавливает то, что она хочет бегать только не по кольцевому маршруту, так как очень не любит считать круги, которые пробежала. Наташа знает основные точки в городе, через которые хотела бы пробежать, а также длины дорог между этими маршрутами, однако из-за того, что город маленький, и полиция не патрулирует все дороги должным образом, дороги могут быть как полностью безопасными, так и быть полностью опасными. Наташа не боится опасных дорог, так как занималась много лет боевыми искусствами, но ей хотелось бы, чтобы её маршрут был максимально безопасным, насколько это возможно, поэтому каждое утро Наташа также слушает новости, прежде чем пойти на пробежку, чтобы понимать, какие точки сегодня являются точно небезопасными (безопасность равна 0). Помогите Наташе построить самый большой некольцевой маршрут, который является самым безопасным.

Входные данные

На первой строке входного файла подаются три целых числа N ($1 \leq N \leq 100$), M ($1 \leq M \leq 1000$), K ($1 \leq K \leq 100$), где N – количество точек маршрута и M – наборы вида (номер начальной точки (от 1), номер конечной точки (от 1), длина маршрута, степень безопасности (вещественное число $[0;1]$, где 0 – самый небезопасный, 1 – самый безопасный)), K – количество точек маршрута, на которых сейчас нельзя точно находиться, потому что полиция там разыскивает преступника. Следует учесть, что дороги неориентированные, поэтому, если задан набор (1 2 10 0.5), то значит, что также задан набор (2 1 10 0.5).

Выходные данные

На выходе необходимо вывести **номера вершин** самого длинного (*с наибольшим количеством точек*) маршрута без циклов (некольцевого) (начиная с вершины с самым меньшим номером), который может быть построен Наташей в её городе, и который не проходит через вершины, которые полиция назвала утром небезопасными, **длину маршрута**, а также **безопасность маршрута (округлив до двух знаков после запятой; если число целое или имеет один знак, выводить ОБЯЗАТЕЛЬНО С НУЛЯМИ (5.00, 1.70))**, при условии, что безопасность маршрута должна быть не менее 0.5. Безопасность маршрута высчитывается как произведение всех значений безопасности каждого пути на маршруте. Если самых больших (с наибольшим количеством точек) некольцевых маршрутов несколько, необходимо вывести самый безопасный.

Ввод	Вывод
5 5 1 1 2 10 0.8 3 2 5 0.4 1 3 4 1.0 3 4 7 1.0 5 4 4 1.0 4	2 1 3 14 0.80
5 5 2 1 2 5 1.0 3 1 2 1.0 3 2 2 0.9 4 3 10 0.1 1 4 10 0.1 5 4	2 1 3 7 1.00

Примечание: гарантируется

- что искомый маршрут существует;
- что могут существовать дороги вокруг одной точки (петли);
- что никакие две точки маршрута не совпадают;
- разделитель дробной части всегда ТОЧКА.

Пример №1

Полиция утром сообщила только об одной небезопасной точке (на которой орудуют преступники) - это точка №4. Поэтому все маршруты через точку №4 сегодня нужно отменить!

Проверочные тесты

Входные данные	Ожидаемый результат
5 5 1 1 2 10 0.8 3 2 5 0.4 1 3 4 1.0 3 4 7 1.0 5 4 4 1.0 4	2 1 3 14 0.80
5 5 2 1 2 5 1.0 3 1 2 1.0 3 2 2 0.9 4 3 10 0.1 1 4 10 0.1 5 4	2 1 3 7 1.00

28 26 6	3 25 14 12 10 21 24 329 0.54
15 9 82 0.88	
26 17 28 0.8	
10 21 80 1.0	
3 28 85 0.78	
25 3 81 0.83	
1 27 46 0.52	
25 12 77 0.71	
13 16 61 0.91	
17 27 94 0.58	
7 23 76 0.6	
17 5 85 0.63	
24 21 48 0.9	
19 20 54 0.82	
21 5 4 0.77	
12 10 32 0.94	
25 14 10 0.82	
26 28 8 0.91	
12 14 78 0.93	
21 19 96 0.78	
16 19 100 0.76	
20 15 33 0.84	
6 23 44 0.64	
13 28 98 0.86	
7 28 94 0.63	
19 7 51 0.82	
27 26 69 0.75	
11	
6	
19	
9	
1	
5	

26 25 11	3 15 10 22 4 205 0.51
9 19 84 0.89	
3 15 25 0.82	
24 8 86 0.91	
2 14 1 0.92	
2 9 1 0.86	
23 22 97 0.78	
15 16 25 0.83	
5 6 3 0.59	
17 25 45 0.56	
15 10 72 0.75	
7 5 55 0.78	
22 4 27 0.96	
10 2 66 0.61	
12 23 6 0.91	
10 22 81 0.87	
8 25 40 0.99	
3 1 88 0.85	
11 26 47 0.78	
24 23 84 0.84	
8 13 88 0.52	
26 16 48 0.75	
20 8 88 0.52	
9 16 57 0.61	
1 2 90 0.62	
11 22 59 0.66	
19	
24	
11	
7	
2	
6	
26	
16	
9	
21	
20	

25 23 4	4 15 18 24 19 186 0.64
20 7 79 0.54	
3 25 87 0.55	
14 7 25 0.72	
10 17 85 0.91	
5 24 63 0.65	
24 22 88 0.64	
24 15 18 0.67	
24 18 51 0.93	
15 18 98 0.81	
22 6 27 0.99	
21 3 23 0.66	
25 10 44 0.7	
24 19 27 0.95	
10 8 27 0.76	
25 25 100 0.6	
15 4 10 0.89	
1 12 91 0.91	
23 12 70 0.67	
19 21 83 0.68	
8 13 92 0.55	
9 15 15 0.6	
10 18 32 0.65	
10 12 57 0.91	
25	
20	
9	
8	

28 28 2	25 9 4 27 11 2 26 340 0.54
14 7 68 0.98	
7 7 57 0.91	
9 25 67 0.75	
21 19 82 0.65	
1 8 98 0.77	
11 2 69 0.91	
15 28 4 0.96	
4 19 6 0.81	
17 3 57 0.92	
27 3 52 0.75	
18 3 25 0.72	
1 23 79 0.96	
7 28 58 0.54	
4 27 50 0.98	
11 16 13 0.76	
20 11 53 0.7	
2 28 86 0.73	
5 18 83 0.93	
9 2 63 0.64	
26 2 48 0.9	
6 23 15 1.0	
15 7 32 0.71	
24 6 25 0.94	
25 22 44 0.53	
15 15 36 0.56	
10 24 9 0.85	
11 27 48 0.93	
9 4 58 0.96	
14	
28	

29 17 6	15 6 17 10 29 144 0.65
6 15 83 0.87	
26 8 30 0.9	
9 9 25 0.64	
13 14 15 0.53	
28 23 78 0.8	
12 26 8 0.96	
14 3 69 0.68	
27 8 85 0.59	
19 18 45 0.62	
18 21 22 0.67	
28 28 2 0.67	
3 24 96 0.95	
13 9 20 0.85	
17 6 6 0.95	
25 22 67 0.93	
10 17 38 0.8	
10 29 17 0.98	
16	
4	
9	
21	
27	
20	

29 28 11	17 9 18 19 159 0.61
18 9 46 0.93	
29 3 78 0.84	
22 5 47 0.64	
16 10 80 0.76	
9 17 58 0.68	
24 6 79 0.63	
15 21 17 0.54	
14 12 33 0.72	
1 25 8 0.96	
5 18 66 0.72	
7 1 18 0.79	
15 15 60 0.6	
29 27 7 0.74	
21 22 36 0.66	
20 23 72 0.76	
16 26 60 0.96	
22 25 14 0.57	
18 19 55 0.97	
22 3 43 0.93	
13 13 79 0.96	
17 1 69 0.65	
8 20 75 1.0	
16 23 90 0.83	
24 24 26 0.97	
12 22 10 0.61	
12 28 62 0.72	
17 28 8 0.51	
15 28 41 1.0	
27	
29	
1	
10	
23	
24	
4	
3	
2	
6	
16	

16 16 8 13 14 74 0.95 2 15 57 0.65 4 6 18 0.52 7 13 25 0.61 16 4 41 0.78 5 8 3 0.92 7 2 79 0.69 2 12 79 0.93 12 15 6 0.84 13 1 76 1.0 14 9 56 0.86 9 15 93 0.62 6 1 25 0.65 12 6 10 0.99 13 11 31 0.7 1 1 55 0.88 16 5 14 7 11 8 10 2	13 1 6 12 15 117 0.54
21 11 4 19 13 9 0.66 18 12 97 0.89 7 3 63 0.76 1 12 81 0.79 6 18 16 0.74 11 18 80 0.64 12 19 17 0.6 6 9 66 0.56 14 5 35 0.77 7 6 55 0.81 19 15 15 0.63 10 15 1 14	7 6 18 12 168 0.53

14 14 2 3 11 65 0.71 11 1 46 0.6 10 12 18 0.72 2 4 90 0.5 2 9 64 0.83 14 4 67 0.84 4 9 90 0.79 1 3 55 0.97 8 1 14 0.57 5 9 32 0.61 9 6 14 0.5 3 4 83 0.97 14 6 41 0.78 2 5 6 0.6 10 6	1 3 4 9 2 292 0.62
10 5 4 10 3 40 0.67 9 8 17 0.75 4 2 85 0.84 4 8 90 0.91 6 9 68 0.91 7 5 1 3	2 4 8 9 6 260 0.52
8 7 1 8 2 46 0.97 1 7 92 1.0 8 4 31 0.78 4 5 8 0.71 2 3 19 0.95 4 6 65 0.71 2 7 22 0.88 1	3 2 8 4 5 104 0.51

29 23 3 28 19 44 0.58 2 6 1 0.59 29 19 70 0.54 9 24 57 1.0 13 27 79 0.63 22 12 60 0.92 8 29 81 0.99 1 26 73 0.77 12 8 16 0.89 23 15 70 0.67 17 3 34 0.87 23 12 37 0.58 8 23 28 0.75 23 28 61 0.51 26 12 54 0.84 28 4 3 0.92 24 19 16 0.79 17 9 37 0.67 17 21 46 0.86 19 17 88 0.69 7 23 86 0.95 17 15 86 0.64 15 4 27 0.71 26 27 10	7 23 8 12 22 190 0.58
20 18 1 20 18 23 0.89 14 7 44 0.55 18 13 69 0.59 11 8 24 0.5 5 11 59 0.98 12 18 33 0.84 16 7 18 0.5 13 12 4 0.78 2 5 40 0.7 17 14 87 0.93 8 1 87 0.77 3 9 94 0.66 12 2 15 0.92 9 18 56 0.6 15 17 17 0.52 20 2 27 0.99 3 13 92 0.79	11 5 2 20 18 12 182 0.51

10 1 22 0.94 17	
29 26 1 26 6 95 0.93 13 23 97 0.56 3 9 76 0.62 15 16 14 0.72 8 2 2 0.52 25 2 85 0.9 12 17 73 0.81 17 14 94 0.73 29 10 8 0.94 24 18 100 0.66 27 17 76 0.64 13 17 8 0.55 3 1 13 0.7 8 24 89 0.5 18 8 6 0.6 25 6 5 0.84 14 26 68 0.94 26 29 42 0.93 21 20 68 0.86 5 2 85 1.0 27 16 4 0.75 9 2 20 0.5 17 23 79 0.63 3 17 4 0.97 26 16 8 0.87 4 16 53 0.53 2	3 17 14 26 29 10 216 0.58

25 21 6	8 17 22 25 5 21 23 244 0.70
24 5 92 0.86	
5 20 72 0.63	
21 5 49 0.99	
23 14 82 0.55	
25 22 3 0.92	
3 12 71 0.78	
6 11 72 0.8	
7 16 12 0.89	
11 3 86 0.99	
8 8 57 0.58	
5 25 15 0.95	
23 6 59 0.61	
23 21 60 0.95	
8 13 23 0.7	
17 21 85 0.81	
16 12 23 0.74	
10 3 64 0.63	
20 8 34 0.51	
22 16 58 0.7	
17 8 64 0.94	
22 17 53 0.9	
10	
14	
20	
1	
11	
19	

25 23 2	6 1 14 2 20 13 8 290 0.50
23 22 49 0.97	
8 3 11 0.71	
10 1 44 0.73	
1 14 78 0.82	
15 2 79 1.0	
14 2 43 0.86	
20 13 46 0.89	
16 25 83 0.76	
10 18 13 0.94	
5 23 94 0.66	
6 1 5 0.87	
16 20 67 0.95	
19 25 9 0.91	
12 18 93 0.9	
19 4 21 0.9	
11 21 19 0.97	
13 4 5 0.75	
5 9 29 0.62	
9 22 85 0.91	
8 13 44 0.99	
20 2 74 0.93	
16 1 71 0.66	
8 21 77 0.63	
17	
19	

27 21 5	4 2 23 17 1 6 320 0.59
17 5 21 0.87	
1 17 45 0.95	
4 26 59 0.73	
10 2 38 0.96	
19 3 65 1.0	
9 5 71 0.73	
12 9 39 0.79	
25 3 74 0.93	
23 17 87 0.92	
20 26 86 0.66	
11 7 50 0.85	
13 8 16 0.89	
10 1 81 0.6	
24 22 42 0.97	
2 23 78 0.75	
17 19 44 0.66	
1 21 71 0.74	
8 27 66 0.54	
22 16 66 0.5	
1 6 80 0.93	
2 4 30 0.97	
5	
8	
22	
25	
10	

28 22 7	2 21 18 28 26 17 319 0.62
2 21 54 0.99	
18 22 64 0.92	
11 1 26 0.7	
20 7 81 0.81	
13 19 94 0.9	
18 21 71 0.82	
20 20 86 0.91	
26 17 67 0.94	
1 13 39 0.85	
10 16 89 0.75	
18 7 99 0.83	
25 23 77 0.63	
26 28 60 0.83	
17 16 4 0.55	
14 7 38 0.99	
4 3 40 0.84	
18 28 67 0.98	
25 1 64 0.65	
15 6 85 0.72	
25 10 93 0.53	
12 13 79 0.51	
16 27 74 0.7	
16	
22	
7	
20	
4	
8	
25	

Пример решения

mxPath = ([], 0, 0)

```
def DFS(v, path, op, ln):
    global mxPath
    if v not in d:
        return
    to = d[v]
    for i in range(len(to)):
        b = to[i]
        if b[0] in path:
            continue
        if b[0] == path[0]:
            continue
```



```

    if b[0] != path[0]:
        if op * b[2] >= 0.5:
            if (len(mxPath[0]) < len(path) + 1) or (len(mxPath[0]) == len(path) + 1 and
abs(mxPath[1] - op*b[2]) > 1e-4 and op*b[2] > mxPath[1]):
                mxPath = (path.copy() + [b[0]], op * b[2], ln + b[1])
                DFS(b[0], path + [b[0]], op * b[2], ln + b[1])

```

```

points = []
n, m, k = map(int, input().split())
d = dict()
edges = []
for i in range(1, m+1):
    edges.append(input())

```

```

dang = []
for i in range(k):
    dang.append(int(input()))

```

```

for i in range(1, m+1):
    s = edges[i-1].split()
    s[0] = int(s[0])
    s[1] = int(s[1])
    if s[0] not in dang and s[1] not in dang:
        d[s[0]] = d.get(s[0], [])
        d[s[1]] = d.get(s[1], [])
        d[s[0]].append((s[1], int(s[2]), float(s[3])))
        d[s[1]].append((s[0], int(s[2]), float(s[3])))

```

```

for k in d:
    d[k].sort()

```

```

color = [0] * (n + 1)

```

```

for i in range(1, n + 1):
    DFS(i, [i], 1, 0)

```

```

print(*mxPath[0], mxPath[2], "%.2lf" % round(mxPath[1], 2))

```