

Задача 1 (8 баллов) id 5137

Условие

Группа бухгалтеров из компании “БУКа”, работая с электронными таблицами захотели 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 число с двумя знаками после запятой, округленным по правилам математики.

Формула может иметь стандартные перечисленные математические знаки: /, -, *.

Очередность операций определяется правилами математики, скобки не используются в формулах.

Далее идёт массив, представляющей из себя таблицу, где по вертикали идут английские буквы от А до Е, по горизонтали 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 “D0”, “E1”.

На выходе программа должна вывести результат в виде числа с двумя знаками после запятой, округленным по правилам математики, результат работы представленной формулы с данной таблицей.

Примечание: формула имеет от 2 до 5 слагаемых.

Пример: имеется формула и таблица (двумерный массив):

C2/A0+D3-E0

@	0	1	2	3	4
A	2	3	45	-7	45
B	-47	32	33	0	-1
C	32	5	6	3	-10

D 23 -43 -10 -50 4

E 43 16 29 33 2

Ячейка C2 имеет значение 6. Ячейка A0 имеет значение 2. Ячейка D3 имеет значение -50. Ячейка E0 имеет значение 43. Получаем: $6/2 + -50 - 43 = -90$

Входные данные:	Выходные данные:
C2/A0+D3-E0 2 3 45 -7 45 -47 32 33 0 -1 32 5 6 3 -10 23 -43 -10 -50 4 43 16 29 33 2	-90.00
B1/E4/A2 -50 0 28 16 -32 -26 24 -46 35 -20 -42 -15 1 -30 35 19 14 -9 47 4 -36 44 -10 19 43	0.02

Проверочные тесты

Входные данные	Ожидаемый результат
C2/A0+D3-E0 2 3 45 -7 45 -47 32 33 0 -1 32 5 6 3 -10 23 -43 -10 -50 4 43 16 29 33 2	-90
B1/E4/A2 -50 0 28 16 -32 -26 24 -46 35 -20 -42 -15 1 -30 35 19 14 -9 47 4 -36 44 -10 19 43	0.02
A1+B2 -41 -14 -45 26 -8 -17 -14 48 -45 21 -49 -11 19 19 -11 21 31 -8 -33 14 22 -25 21 -30 -42	34
C1-E2/C4+D3-A0 39 34 -9 -31 9 -13 49 -42 -36 -37 -38 25 -26 -31 -41 5 41 43 49 -37 18 -5 -42 -27 -12	33.98
C4-E4/C1+D0 -30 -17 36 -43 46 -40 -35 -2 10 30 10 -8 -35 -46 31 -27 21 30 38 -41 -32 8 42 -48 -50	-2.25
A4+A4+D1/C0 42 12 -18 -8 -27 40 -38 47 -47 15 26 -3 -33 15 -49 -23 -40 -2 2 -10 30 -10 24 39 -32	-55.54

A4+A4+A4+A4+A4 21 11 24 -27 19 45 23 12 40 9 -4 43 26 10 -47 -23 -7 1 37 -47 8 -34 -44 -27 22	95
A0/C2+B1-D4+E3 50 -22 34 1 -14 -31 44 -6 7 -3 -15 25 -45 -40 35 18 -26 -22 -10 32 0 -24 33 8 -22	18.89

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[0])][int(c[1])])
        c = ""
        s += let
s += str(mass[int(c[0])][int(c[1])])
print(round(eval(s), 2))
```

Задача 2 (12 баллов) id 5139

Условие

IT - company “ЕнК” решила создать программу, которая помогала бы пользователям мгновенно трансформировать текст, ошибочно написанный не на той раскладке языка. Чаще всего такая проблема возникает с написанием английских слов на русской раскладке клавиатуры.

На вход программы поступает строка, состоящая из кодов русских символов по таблице ASCII нижнего регистра от а до я. Длина строки $5 \leq X \leq 300$. В предложениях не используются знаки препинания.

На выходе программа должна выдать строку с трансформированными символами с английской раскладки на русскую. Символ пробела остаётся пробелом и не переводится в кодировку.

Например: “\u0430\u043a\u0448\u0443\u0442\u0432\u044b \u0444\u043a\u0443\u0435\u0440\u0449\u044b\u0443 \u0446\u0440\u0449\u0444\u0434\u0446\u0444\u043d\u044b \u044b\u0433\u0437\u0437\u0449\u043a\u0435\u0433\u044b”, должно преобразоваться в “friends are those who always support us?”

Входные данные:

Выходные
данные:

\u0430\u043a\u0448\u0443\u0442\u0432\u044b \u0444\u043a\u0443\u0435\u0440\u0449\u044b\u0443 \u0446\u0440\u0449\u0444\u0434\u0446\u0444\u043d\u044b \u044b\u0433\u0437\u0437\u0449\u043a\u0435 \u0433\u044b

friends are
those who
always
support us

\u043a\u0443\u0444\u0432\u0448\u0442\u043f
\u0438\u0449\u0449\u043b\u044b \u0440\u0443\u0434\u0437\u044b
\u0435\u0449 \u0432\u0443\u043c\u0443\u0434\u0434\u0449\u0437
\u0449\u0433\u043a
\u0448\u044c\u0444\u043f\u0448\u0442\u0444\u0435\u0448\u0449\u0442

reading
books helps
to develop
our
imagination

Проверочные тесты

Входные данные	Ожидаемый результат
\u0430\u043a\u0448\u0443\u0442\u0432\u044b \u0444\u043a\u0443\u0435\u0440\u0449\u044b\u0443 \u0446\u0440\u0449\u0444\u0434\u0446\u0444\u043d\u044b \u044b\u0433\u0437\u0437\u0449\u043a\u0435 \u0433\u044b	friends are those who always support us
\u043a\u0443\u0444\u0432\u0448\u0442\u043f \u0438\u0449\u0449\u043b\u044b \u0440\u0443\u0434\u0437\u044b \u0435\u0449 \u0432\u0443\u043c\u0443\u0434\u0449\u0437	reading books helps to develop

\u0449\u0433\u043a \u0448\u044c\u0444\u043f\u0448\u0442\u0444\u0435\u0448\u0449\u0442 2	our imagination
u0440\u0449\u044c\u0443\u0446\u0446\u0449\u043a\u043b \u0435\u0443\u0444\u0441\u0441\u0441\u0440\u0443\u044b \u0433\u044b \u0435\u0449 \u0438\u0443 \u0449\u043a\u043f\u0444\u0442\u0448\u044f\u0443\u0432 \u0444\u0442\u0432 \u043a\u0443\u044b\u0437\u0449\u0442\u044b\u0448\u0438\u0434\u0443 3	homework teaches us to be organized and responsible
u0438\u043a\u0443\u0444\u0448\u043b\u0430\u0444\u044b\u0435 \u043f\u0448\u043c\u0443\u044b \u0433\u044b \u0443\u0442\u0443\u043a\u043f\u043d \u0430\u0449\u043a \u0435\u0440\u0443 \u0446\u0440\u0449\u0434\u0443 \u0432\u0444\u043d	breakfast gives us energy for the whole day
u044b\u0437\u0449\u043a\u0435 \u044c\u0444\u0448\u043b\u0443\u044b \u0449\u0433\u043a \u0438\u0449\u0432\u043d \u044b\u0435\u043a\u0449\u0442\u043f \u0444\u0442\u0432 \u0440\u0443\u0444\u0434\u0435\u0440\u043d	sport makes our body strong and healthy
u0444 \u044b\u044c\u0448\u0448\u0434\u0443\u0443 \u0441\u0441\u0442\u0442 \u0441\u0440\u0444\u043f\u0443 \u0449\u0433\u043a \u044c\u0449\u0449\u0432 \u0444\u0442\u0432 \u0435\u0440\u0443 \u0449 \u0449\u0435\u0440\u0443\u0443\u043a\u044b \u0430\u0449\u043a \u0435\u0440\u0443 \u0443\u0438\u0435\u0435\u0443\u043a	a smile can change our mood and that of others for the better
u0430\u0444\u044c\u0448\u0448\u0434\u043d \u0448\u044b \u0435\u0440\u0443 \u0430\u0449\u0433\u0442\u0443\u0448\u0435\u0448\u0449\u0442 \u0449\u0430 \u0440\u0444\u0437\u0437\u0448\u0442\u0443\u044b\u044b \u0444\u0442\u0432 \u044b\u0433\u0441\u0441\u0443\u044b\u044b	family is the foundation of our happiness and success
u0442\u0444\u0435\u0433\u043a\u0443 \u0448\u044b \u0444 \u044b\u0449\u0433\u043a\u0441\u0441 \u0449\u0430 \u0448\u0442\u0448\u043a\u0444\u0435\u0448\u0443\u044b\u044b \u0444\u0442\u0432 \u044b\u0433\u0441\u0441\u0443\u044b\u044b	nature is a source of inspiration and relaxation

Пример решения

import unicodedata

```
d = {
    "a": "f",
```

```

"б": "<",
"в": "d",
"г": "u",
"д": "l",
"е": "t",
"ё": "´",
"ж": ";",
"з": "p",
"и": "b",
"й": "q",
"к": "r",
"л": "k",
"м": "v",
"н": "y",
"о": "j",
"п": "g",
"р": "h",
"с": "c",
"т": "n",
"у": "e",
"ф": "a",
"х": "[",
"ц": "w",
"ч": "x",
"ш": "i",
"щ": "o",
"ъ": "]",
"ы": "s",
"ь": "m",
"э": "'",
"ю": ". ",
"я": "z",
}

```

```

s = input()[1:]
s = s.split("\\u")

```

```

res = ""
for el in s:
    if el == " ":
        res += " "
    else:
        res += d[chr(int(el, 16))]
        if " " in el:
            res += " " * el.count(" ")
print(res)

```

Задача 3 (16 баллов) id 5148

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 прямые рейсы, поэтому он решил полететь с пересадками. Алексей выписал все возможные рейсы, которые могут ему пригодиться, включая информацию о том, откуда летит, куда летит, в какое время по Московскому времени они вылетают, через какое время прилетают. Алексей точно знает, что на пересадку с одного рейса на другой требуется минимум 1.5 часа, поэтому, если следующий рейс отправляется меньше, чем через 1.5 часа, то он на него не успевает и начинает ждать другой. Помогите Алексею определить, сможет ли он добраться до своей бабушки или нет, притом за самое минимальное время, которое не превышает 48 часов. Если не сможет, то вывести 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 перелёта 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 рейсы летают **ЕЖЕДНЕВНО**.

Выходные данные:

Выведите на первой строке маршрут, который сможет пролететь Алексей, на второй строке время в пути. Или вывести 0, если маршрута нет.

Ввод	Вывод
4 Moscow Saint Petersburg 10:15 1:15 Vladivostok Novosibirsk 12:00 5:50 Saint Petersburg Novosibirsk 13:00 5:10 Novosibirsk Malokribirsk 20:00 3:05	Moscow Saint Petersburg Novosibirsk Malokribirsk 12:50

4 Moscow Saint Petersburg 10:15 1:15 Vladivostok Novosibirsk 12:00 5:50 Saint Petersburg Novosibirsk 12:00 5:10 Novosibirsk Malokribirsk 20:00 3:05	Moscow Saint Petersburg Novosibirsk Malokribirsk 36:50
---	--

Пример №2

Алексей может отправиться из Москвы только в Санкт-Петербург. Отправляется он в 10:15, летит 1 час 15 минут. Значит прибудет в 11:30. Самолёт из Санкт-Петербурга только единственный, он отправляется в 12:00, значит Алексей не успеет на него, так как на пересадку только 30 минут. Поэтому Алексею придётся ждать следующего самолёта 24 часа и 30 минут, так как известно, что рейсы летают ежедневно. Поэтому дальше он сможет продолжить путь в Новосибирск, а оттуда в Малокрибирск и уложиться в 48 часов.

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 1:15 Vladivostok Novosibirsk 12:00 5:50 Saint Petersburg Novosibirsk 13:00 5:10 Novosibirsk Malokribirsk 20:00 3:05	Moscow Saint Petersburg Novosibirsk Malokribirsk 12:50
4 Moscow Saint Petersburg 10:15 1:15 Vladivostok Novosibirsk 12:00 5:50 Saint Petersburg Novosibirsk 12:00 5:10 Novosibirsk Malokribirsk 20:00 3:05	Moscow Saint Petersburg Novosibirsk Malokribirsk 36:50

15 Saint Petersburg Volgograd 23:20 05:29 Vorkuta Vladikavkaz 21:55 04:41 Saint Petersburg Nalchik 14:39 01:08 Moscow Samara 07:10 07:48 Vladikavkaz Malokribirsk 04:34 01:50 Vorkuta Vladikavkaz 07:04 01:51 Volgograd Saransk 07:01 08:55 Naberezhnye Chelny Vorkuta 04:45 03:21 Severodvinsk Nalchik 04:01 04:56 Murom Vladikavkaz 00:44 03:44 Vorkuta Salekhard 21:40 03:09 Samara Naberezhnye Chelny 17:21 08:57 Malokribirsk Saransk 13:00 08:39 Vorkuta Saransk 14:49 09:35 Volgograd Vologda 05:26 04:19	Moscow Samara Naberezhnye Chelny Vorkuta Vladikavkaz Malokribirsk 47:14
14 Saratov Samara 17:29 05:22 Saratov Volgograd 22:01 01:23 Moscow Vladimir 16:48 01:08 Salekhard Nadym 05:34 00:56 Vorkuta Saint Petersburg 17:06 09:27 Moscow Murom 11:04 05:03 Vladimir Saransk 12:02 02:22 Saransk Salekhard 22:52 08:57 Volgograd Nadym 15:10 03:15 Severodvinsk Saint Petersburg 13:59 08:59 Vorkuta Vladikavkaz 08:29 07:54 Salekhard Saratov 02:36 07:15 Salekhard Vologda 18:44 01:30 Salekhard Malokribirsk 10:53 05:45	Moscow Vladimir Saransk Salekhard Malokribirsk 47:50

15 Nalchik Salekhard 04:49 08:09 Salekhard Severodvinsk 23:12 00:56 Vologda Vorkuta 05:47 02:19 Nadym Vorkuta 23:02 01:03 Naberezhnye Chelny Saransk 17:23 06:26 Severodvinsk Malokribirsk 18:51 04:16 Saratov Saint Petersburg 14:11 05:04 Samara Nalchik 00:47 03:35 Salekhard Severodvinsk 21:44 05:00 Murom Severodvinsk 12:51 04:44 Murmansk Saratov 17:56 05:25 Vladikavkaz Salekhard 16:00 02:52 Samara Severodvinsk 11:35 07:40 Moscow Vladikavkaz 00:12 09:45 Vorkuta Naberezhnye Chelny 22:48 02:26	Moscow Vladikavkaz Salekhard Severodvinsk Malo kribirsk 46:55
14 Saratov Vladikavkaz 19:19 06:05 Naberezhnye Chelny Saransk 18:29 01:19 Saransk Nalchik 00:26 02:14 Malokribirsk Salekhard 15:58 00:58 Moscow Vorkuta 20:29 02:36 Saratov Salekhard 01:16 05:32 Volgograd Nadym 19:35 01:41 Saratov Vorkuta 00:17 03:55 Nalchik Malokribirsk 05:40 09:40 Ryazan Saint Petersburg 10:04 07:40 Vorkuta Volgograd 03:48 04:41 Ryazan Vologda 21:22 05:17 Murmansk Vorkuta 23:42 08:00 Nadym Malokribirsk 13:05 04:18	Moscow Vorkuta Volgograd Nadym Malokribirsk 44:54

<p>11</p> <p>Moscow Saint Petersburg 07:13 07:23</p> <p>Saint Petersburg Vladimir 06:43 05:48</p> <p>Vladimir Samara 04:21 07:59</p> <p>Vladikavkaz Malokribirsk 16:50 06:38</p> <p>Severodvinsk Nadym 09:05 06:07</p> <p>Nadym Saint Petersburg 11:10 05:56</p> <p>Naberezhnye Chelny Saransk 04:49 01:11</p> <p>Saint Petersburg Volgograd 19:49 03:12</p> <p>Moscow Vorkuta 17:54 00:41</p> <p>Volgograd Vladikavkaz 02:19 01:54</p> <p>Moscow Ryazan 12:15 06:47</p>	<p>Moscow Saint Petersburg Volgograd Vladikavkaz Malokribirsk 40:15</p>
<p>15</p> <p>Moscow Saratov 12:15 03:22</p> <p>Saratov Volgograd 19:32 05:53</p> <p>Naberezhnye Chelny Saint Petersburg 07:39 01:28</p> <p>Samara Malokribirsk 05:54 02:00</p> <p>Volgograd Samara 03:53 09:15</p> <p>Murom Vorkuta 22:58 03:02</p> <p>Vladimir Nadym 19:04 05:35</p> <p>Vladimir Naberezhnye Chelny 17:42 04:22</p> <p>Murom Saint Petersburg 06:21 09:06</p> <p>Severodvinsk Vologda 05:56 02:33</p> <p>Saint Petersburg Naberezhnye Chelny 03:21 01:50</p> <p>Saratov Samara 02:16 04:23</p> <p>Saint Petersburg Vorkuta 07:29 02:31</p> <p>Vorkuta Vologda 08:11 04:10</p> <p>Vladikavkaz Moscow 11:13 01:01</p>	<p>Moscow Saratov Volgograd Samara Malokribirsk 43:39</p>

12 Moscow Vladimir 12:05 03:22 Saransk Severodvinsk 07:52 02:09 Saint Petersburg Samara 18:01 03:20 Naberezhnye Chelny Murom 06:26 07:37 Vorkuta Salekhard 15:02 08:12 Saint Petersburg Vladikavkaz 08:57 06:06 Murom Malokribirsk 16:00 01:49 Ryazan Saransk 13:43 08:27 Saratov Moscow 19:24 00:33 Moscow Murom 16:19 01:25 Moscow Murmansk 14:11 04:50 Vladimir Naberezhnye Chelny 18:57 02:36	Moscow Murom Malokribirsk 25:30
15 Saratov Vologda 15:23 01:01 Saransk Severodvinsk 18:28 02:44 Severodvinsk Samara 04:05 02:52 Naberezhnye Chelny Vologda 10:29 07:15 Vorkuta Saransk 14:40 03:29 Vorkuta Samara 08:59 06:32 Moscow Saransk 12:18 02:42 Vladikavkaz Severodvinsk 09:23 08: 17 Volgograd Vorkuta 07:16 08:14 Samara Naberezhnye Chelny 15:18 05:52 Samara Malokribirsk 05:32 02:28 Volgograd Murom 21:11 04:28 Naberezhnye Chelny Nadym 02:56 04:51 Nalchik Malokribirsk 00:40 06:53 Murom Saratov 17:26 02:38	Moscow Saransk Severodvinsk Samara Malokribirsk 43:42

13 Moscow Severodvinsk 07:24 06:18 Nalchik Volgograd 10:33 01:11 Salekhard Naberezhnye Chelny 18:40 01:12 Rybinsk Murmansk 20:04 01:36 Saransk Salekhard 18:52 06:26 Volgograd Murmansk 05:22 03:51 Saint Petersburg Malokribirsk 17:44 08:59 Ryazan Naberezhnye Chelny 03:04 03:01 Samara Murom 06:33 08:16 Saransk Nalchik 20:25 03:59 Moscow Saransk 09:47 03:00 Volgograd Malokribirsk 21:07 09:55 Vorkuta Samara 21:53 06:37	Moscow Saransk Nalchik Volgograd Malokribirsk 45:15
12 Volgograd Salekhard 20:34 02:00 Saransk Volgograd 13:09 03:44 Murom Ryazan 18:37 03:10 Murom Nalchik 15:04 07:06 Nalchik Malokribirsk 23:42 02:15 Saint Petersburg Severodvinsk 05:42 08:1 2 Vladikavkaz Murom 02:30 06:09 Moscow Vladikavkaz 06:25 08:21 Salekhard Nalchik 12:19 06:15 Volgograd Murmansk 08:26 09:54 Saint Petersburg Vologda 09:56 09:23 Vladimir Salekhard 10:08 05:17	Moscow Vladikavkaz Murom Nalchik Malokribirsk 43:32

12 Vorkuta Vologda 15:28 00:31 Moscow Severodvinsk 05:09 01:16 Malokribirsk Salekhard 12:36 07:41 Saratov Volgograd 11:03 07:18 Saratov Volgograd 20:19 05:21 Saint Petersburg Nalchik 01:14 05:45 Ryazan Severodvinsk 10:28 05:46 Saratov Vladikavkaz 10:25 04:14 Salekhard Vorkuta 23:16 07:23 Saransk Malokribirsk 23:04 04:20 Vladikavkaz Saransk 01:49 01:55 Severodvinsk Vladikavkaz 09:27 06:22	Moscow Severodvinsk Vladikavkaz Saransk Malokribirsk 46:15
15 Naberezhnye Chelny Murmansk 19:42 09:15 Rybinsk Naberezhnye Chelny 01:07 02:16 Ryazan Rybinsk 02:08 05:23 Moscow Salekhard 10:16 05:02 Saransk Murom 11:39 06:11 Saransk Vologda 03:32 08:34 Saratov Salekhard 13:49 05:15 Malokribirsk Saransk 18:12 03:13 Ryazan Volgograd 03:12 02:01 Vorkuta Samara 18:57 01:02 Naberezhnye Chelny Malokribirsk 09:24 01:12 Salekhard Vladimir 22:38 06:57 Nalchik Murom 04:55 01:48 Moscow Ryazan 18:17 02:50 Saransk Nadym 17:36 01:26	Moscow Ryazan Rybinsk Naberezhnye Chelny Malokribirsk 40:19

14 Saransk Saint Petersburg 19:31 08:59 Naberezhnye Chelny Ryazan 02:23 09:49 Moscow Vorkuta 02:57 07:23 Samara Vologda 17:12 07:16 Ryazan Saint Petersburg 19:04 03:09 Saratov Vladikavkaz 19:42 07:54 Ryazan Malokribirsk 21:03 06:54 Naberezhnye Chelny Vladimir 02:54 02:09 Moscow Samara 07:12 00:57 Samara Naberezhnye Chelny 12:16 05:36 Severodvinsk Saransk 08:12 07:59 Nadym Saratov 16:43 05:13 Ryazan Vologda 00:51 05:17 Severodvinsk Murom 14:19 04:49	Moscow Samara Naberezhnye Chelny Ryazan Malokribirsk 44:45
8 Rybinsk Saratov 16:35 09:16 Rybinsk Malokribirsk 03:21 02:59 Moscow Murmansk 08:39 01:42 Vologda Vladikavkaz 17:34 09:51 Severodvinsk Rybinsk 23:47 01:11 Saratov Nalchik 06:31 01:38 Saransk Salekhard 11:36 09:30 Murmansk Severodvinsk 22:56 08:13	Moscow Murmansk Severodvinsk Rybinsk Malokribirsk 45:41
14 Moscow Ryazan 07:57 01:25 Murmansk Saint Petersburg 16:40 00:51 Vologda Naberezhnye Chelny 23:13 07:16 Saint Petersburg Vladimir 20:28 09:32 Malokribirsk Murom 14:39 06:36 Vologda Moscow 03:09 00:44 Saransk Vladimir 13:35 09:07 Nadym Ryazan 12:21 05:54 Rybinsk Murmansk 12:53 02:11	Moscow Ryazan Severodvinsk Naberezhnye Chelny Malokribirsk 34:32

Vologda Saratov 03:53 08:03 Naberezhnye Chelny Malokribirsk 17:43 00:46 Ryazan Severodvinsk 19:00 02:43 Samara Saint Petersburg 18:40 05:52 Severodvinsk Naberezhnye Chelny 02:46 09:41	
15 Murmansk Nadym 03:34 08:53 Naberezhnye Chelny Volgograd 09:03 01:39 Ryazan Murom 13:31 06:40 Vorkuta Rybinsk 15:22 05:11 Salekhard Malokribirsk 06:25 02:09 Moscow Vologda 06:03 02:01 Saransk Vorkuta 19:50 07:30 Vologda Saransk 12:23 00:31 Moscow Saratov 04:03 01:56 Vorkuta Malokribirsk 12:11 02:08 Severodvinsk Samara 06:46 04:40 Saratov Vladikavkaz 02:40 01:35 Vladimir Samara 17:29 05:58 Vorkuta Salekhard 06:03 05:09 Saratov Vorkuta 23:58 02:30	Moscow Vologda Saransk Vorkuta Malokribirsk 32:16
14 Murmansk Salekhard 20:02 05:33 Salekhard Ryazan 09:53 08:06 Moscow Salekhard 01:10 02:37 Nadym Vologda 14:53 08:27 Vladimir Moscow 06:22 07:09 Saratov Moscow 00:31 08:09 Volgograd Murom 13:43 06:08 Ryazan Rybinsk 23:40 05:47 Murmansk Ryazan 15:38 08:29 Rybinsk Severodvinsk 04:00 03:12 Severodvinsk Samara 01:01 00:50 Nadym Vladikavkaz 01:18 02:36 Saratov Saransk 12:54 06:18 Rybinsk Malokribirsk 09:26 05:47	Moscow Salekhard Ryazan Rybinsk Malokribirsk 38:03

15 Vologda Vorkuta 23:24 05:29 Saratov Vologda 17:21 02:42 Malokribirsk Murom 14:21 04:44 Naberezhnye Chelny Samara 21:47 04:58 Naberezhnye Chelny Vladimir 04:52 03:01 Vladikavkaz Nalchik 13:21 08:23 Murom Saratov 12:13 09:20 Murom Saint Petersburg 03:21 03:22 Moscow Saint Petersburg 16:38 01:46 Murom Nadym 22:03 07:25 Naberezhnye Chelny Samara 03:11 05:17 Vladimir Salekhard 04:53 02:15 Vologda Malokribirsk 04:16 00:33 Saransk Murom 21:28 03:27 Moscow Murom 08:56 06:20	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 >= 90:
            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
routes = {}
for _ in range(n):
    rout = input().split('|')
    if rout[0] in routes:
        h, m = [int(i) for i in rout[2].split(':')]
        h1, m1 = [int(i) for i in rout[3].split(':')]
        routes[rout[0]] = routes[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(':')]
        routes[rout[0]] = [[rout[1], h * 60 + m, h1 * 60 + m1]]

```

```

if 'Moscow' in routes:
    for t in routes['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] <= 2880:
    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 5141

Условие

Пете на день рождения подарили Робота, однако робот оказался таким, что его нужно обучать. Петя решил обучить робота находить на клеточном поле самую большую равнобедренную трапецию направленную вверх и выводить её 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 самую большую равнобедренную трапецию направленную вверх из пустых клеток, а также выводить её площадь. Петя гарантирует, что в его матрице такая трапеция существует 100%. Но важно также учесть, что трапеция должка быть **ОБЯЗАТЕЛЬНО** ограничена 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 трапеции, которая есть на данном клеточном поле (целое число без запятой или точки!). На второй строке требуется вывести координаты левой нижней клетки начала трапеции через пробел в формате (номер_строки номер_столбца (нумерация с ЕДИНИЦЫ)).

Ввод	Вывод
4 10 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 0 0 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1	6 3 2

7 15 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 0 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 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1	15 4 8

Примечание:

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

Пример №1

```

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

```

Трапеция, ограниченная единицами здесь только одна, поэтому её площадь и максимальна.

Проверочные тесты

Входные данные	Ожидаемый результат
4 10 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 0 0 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1	6 3 2
7 15 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1	15 4 8
7 15 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 0 0 0 1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1	6 3 2

[illegible]

4 16

[illegible]

32

[illegible]

142

[illegible]

Пример решения

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

vector<pair<int, int>> road;
bool flag = false;

void dfs(int i, int j, vector<vector<char>>& table, vector<vector<bool>>& used) {
    if (used[i][j]) return;
    used[i][j] = true;
    if ((i == 0) || (i == table.size() - 1) || (j == 0) || (j == table[0].size() - 1)) {
        flag = true;
    }
    if (table[i][j] == '1') return;
    road.push_back({ i, j });
    dfs(i + 1, j, table, used);
    dfs(i - 1, j, table, used);
    dfs(i, j + 1, table, used);
    dfs(i, j - 1, table, used);
}

void solve() {
    int N, M; cin >> N >> M;
    vector<vector<char>> table(N + 2, vector<char>(M + 2, '1'));
    for (int i = 1; i <= N; i++) {
        for (int j = 1; j <= M; j++) {
            cin >> table[i][j];
        }
    }
    long long max_result = 0;
    pair<int, int> left_point;
    vector<vector<bool>> used(N + 2, vector<bool>(M + 2, false));
    for (int i = 1; i <= N; i++) {
        for (int j = 1; j <= M; j++) {
            flag = false;
            road.clear();
            if (!used[i][j]) {
                dfs(i, j, table, used);
            }
            else continue;
            if (flag || road.size() == 0) continue;
            sort(road.begin(), road.end());
            pair<int, int> left = road[0];
            int i_left = 0;
            int i_right = 0;
            for (int k = 1; k < road.size(); k++) {
```

```

        if (road[k].first != road[0].first) {
            i_right = k-1;
            break;
        }
    }
    if (i_right == road.size() - 1) continue;
    while (i_right < road.size()) {
        int arr = i_left;
        i_left = i_right + 1;
        i_right += i_right - arr + 3;
        if (i_right >= road.size()) continue;
        bool flag = false;
        for (int k = i_left; k <= i_right; k++) {
            if (road[i_left].first != road[k].first || road[k].second - road[i_left].second != k -
i_left) {
                flag = true;
                break;
            }
        }
        if (flag) break;
    }
    if (i_left != road.size()) continue;
    if (max_result < road.size()) {
        max_result = road.size();
        left_point = road[i_left + i_left - i_right + 1];
    }
}
}
cout << max_result << "\n";
cout << left_point.first << " " << left_point.second;
}

int main()
{
    ios_base::sync_with_stdio(false); cin.tie(nullptr); cout.tie(nullptr);
    solve();
    return 0;
}

```

Задача 5 (20 баллов) id 5145

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 четырёхугольный 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 выпуклый четырёхугольник в вершинах которого находится по одной берёзе, а внутри него содержалось максимальное количество других берёз, а по каждой стороне вне выпуклого четырёхугольника 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

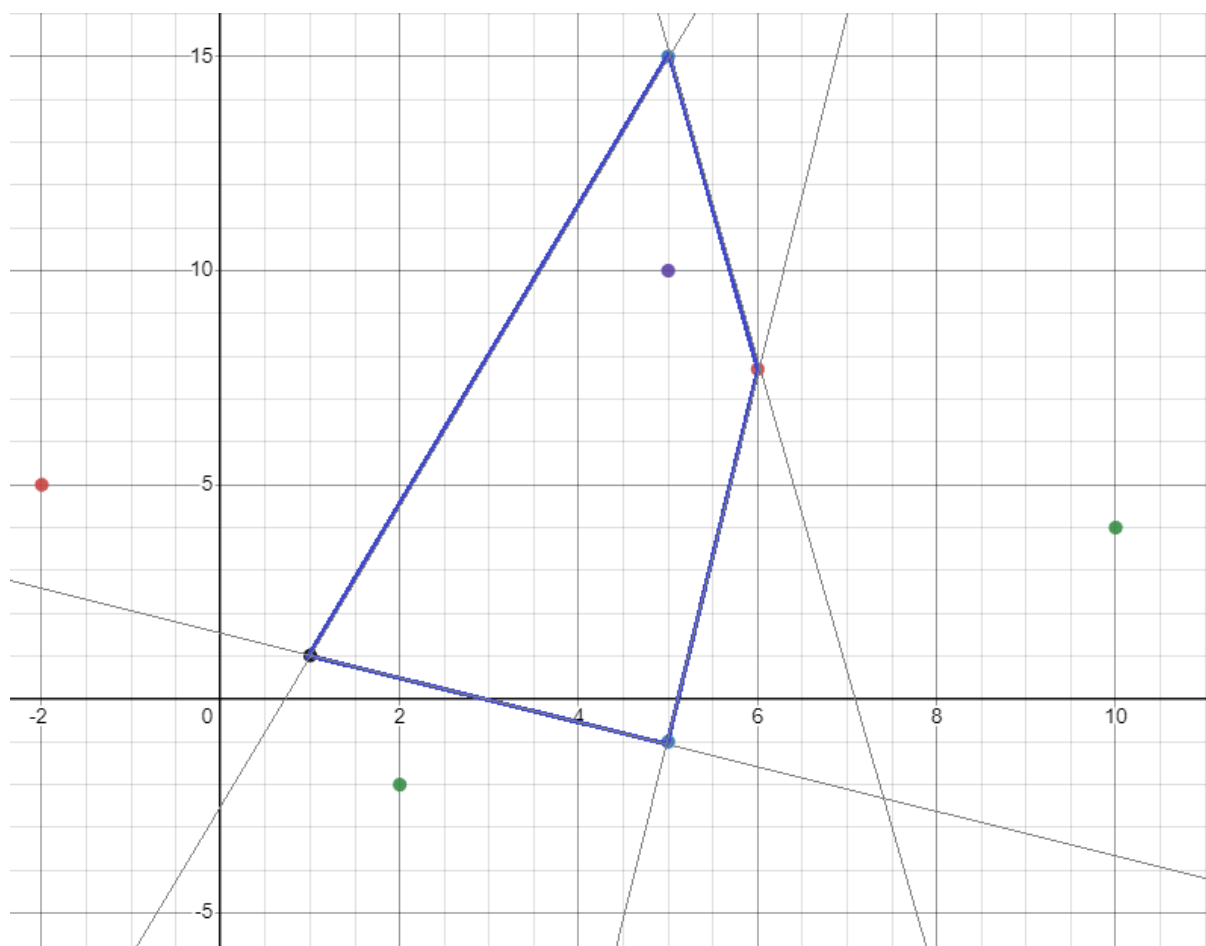
На четырёх строках вывести координаты деревьев (округлив до двух знаков; если значение целое, то выводить ВСЕГДА два знака (например 5.00, либо 1.70)), которые образуют искомые координаты выпуклого четырёхугольного сквера (его вершины), начиная от вершины с максимальным Y и минимальным X , а далее против 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
------	-------

8	5.00 15.00
6.00 7.70	1.00 1.02
5.00 15.00	5.00 -1.00
10.00 4.00	6.00 7.70
5.00 10.00	
1.00 1.0175	
-2.00 5.00	
5.00 -1.00	
2.00 -2.00	
5	0
31.914 0.6194	28.09 0.62
31.1829 -1.6306	28.82 -1.63
30.00 2.00	30.00 2.00
28.8171 -1.6306	31.18 -1.63
28.086 0.6194	31.91 0.62

Пример 1.

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



Проверочные тесты

Входные данные	Ожидаемый результат
8 6.00 7.70 5.00 15.00 10.00 4.00 5.00 10.00 1.00 1.0175 -2.00 5.00 5.00 -1.00 2.00 -2.00	5.00 15.00 1.00 1.02 5.00 -1.00 6.00 7.70
5 31.914 0.6194 31.1829 -1.6306 30.00 2.00 28.8171 -1.6306 28.086 0.6194	0 28.09 0.62 28.82 -1.63 30.00 2.00 31.18 -1.63 31.91 0.62

14 449.0 161.0 -433.5 362.0 -14.0 -259.0 -386.0 -149.5 -462.5 140.5 -67.0 -164.5 -454.0 -284.0 451.0 -353.5 430.0 10.5 -345.0 13.0 305.5 92.0 -136.5 -63.0 143.5 134.0 -57.5 280.0	449.00 161.00 -462.50 140.50 -345.00 13.00 -14.00 -259.00
14 468.5 -78.5 341.5 342.5 -43.0 420.0 399.0 247.0 -453.5 -4.5 -329.5 357.5 -40.5 332.0 -97.0 -74.5 39.0 -465.5 -12.0 63.0 -392.5 378.0 -271.5 428.5 43.0 238.5 -177.5 -128.0	-392.50 378.00 -97.00 -74.50 399.00 247.00 -40.50 332.00

18 402.5 311.0 -266.0 -307.5 -271.0 0.5 -391.5 -398.0 171.0 -408.0 -100.0 460.0 -24.0 -86.0 179.5 -302.0 357.5 -492.0 -433.0 147.0 25.5 -194.0 355.5 294.0 233.0 -313.5 -2.5 63.5 -87.0 458.5 -198.5 -182.5 -225.5 468.0 164.5 -247.0	-2.50 63.50 -271.00 0.50 25.50 -194.00 233.00 -313.50
16 -258.0 -152.0 -71.5 429.0 212.5 2.0 -158.0 -143.0 297.5 -191.5 -386.0 -474.0 345.0 46.5 482.5 -370.5 -172.5 175.5 233.5 275.5 185.5 -199.5 171.0 236.5 -202.0 -160.5 251.0 82.5 -426.0 -460.0 273.0 106.0	233.50 275.50 -172.50 175.50 -426.00 -460.00 482.50 -370.50

18 -207.0 381.5 -354.5 -165.0 -451.5 422.5 -116.5 -234.0 -433.5 -372.5 498.5 -486.0 210.5 -375.0 -232.5 -350.5 128.5 -414.0 -218.5 386.0 -450.0 19.0 -230.0 -330.0 -444.0 -162.5 221.0 -410.0 424.5 332.5 221.5 -321.0 255.0 477.0 -277.5 308.5	-207.00 381.50 -444.00 -162.50 221.00 -410.00 210.50 -375.00
15 488.0 352.0 -308.0 -427.0 379.5 -463.5 372.0 182.0 80.5 391.5 78.5 -271.5 270.0 354.5 340.5 301.0 375.5 -47.0 281.5 323.5 9.5 -394.5 -400.0 -47.0 -458.0 313.5 195.0 278.5 172.5 257.0	270.00 354.50 -400.00 -47.00 379.50 -463.50 372.00 182.00

16 -249.5 -215.5 22.0 -300.0 451.5 199.5 -358.0 -430.5 -14.0 -462.0 -57.0 -489.0 105.5 221.5 460.0 434.0 -352.0 101.5 122.0 -130.0 -2.0 279.0 -319.0 -121.5 -32.5 -250.0 -378.5 50.5 -1.5 -105.0 37.5 86.5	460.00 434.00 -352.00 101.50 -358.00 -430.50 -14.00 -462.00
12 -204.5 -75.0 -317.5 -498.0 460.0 339.0 155.0 -422.0 -494.0 -152.5 452.0 69.0 -157.0 -31.0 -268.0 416.0 313.5 -54.5 -6.0 -417.5 36.5 -330.5 -406.5 131.5	-268.00 416.00 -494.00 -152.50 -6.00 -417.50 452.00 69.00
11 -141.5 -57.5 98.0 -9.5 -115.0 -479.5 -173.5 -438.0 144.0 -481.5 459.5 -223.5 472.0 63.5 276.0 170.5 -190.0 -205.0 286.0 160.5 298.5 360.5	298.50 360.50 -190.00 -205.00 -115.00 -479.50 459.50 -223.50

18 146.5 109.0 455.0 290.5 353.5 -93.0 -384.0 345.5 -451.5 -190.0 -212.0 -111.5 367.0 463.0 -123.0 -460.5 491.5 173.5 380.5 57.5 4.0 -292.5 -428.0 394.5 -399.5 -135.0 153.0 -59.5 132.0 256.5 427.0 -242.5 50.0 -261.5 -101.5 -136.5	-428.00 394.50 -123.00 -460.50 380.50 57.50 491.50 173.50
14 426.0 -113.0 373.5 -293.5 -144.0 -319.5 -156.5 132.0 -379.5 308.0 -336.0 101.5 199.5 -324.5 229.0 -355.0 70.5 -113.5 -250.0 481.0 174.5 -118.0 228.5 -452.0 -318.0 -111.5 -361.0 -272.5	-250.00 481.00 -361.00 -272.50 229.00 -355.00 174.50 -118.00

12 439.0 276.0 359.0 118.0 362.0 202.5 -395.0 117.5 -217.5 18.0 403.5 -109.0 -153.5 35.5 469.0 -204.0 47.5 -474.5 -438.0 -199.5 148.0 19.5 422.5 -33.5	439.00 276.00 -153.50 35.50 -438.00 -199.50 403.50 -109.00
13 -152.5 70.0 324.5 393.5 109.0 -230.0 -403.5 -106.0 188.5 -323.5 -140.0 -117.5 -210.0 -387.0 466.0 -477.0 486.5 209.0 -74.5 -457.5 439.5 -392.5 184.0 459.5 208.5 425.5	208.50 425.50 -140.00 -117.50 188.50 -323.50 439.50 -392.50
17 26.5 -74.5 361.0 208.0 437.0 -305.5 207.0 279.0 -192.5 9.0 -498.5 -340.5 -27.0 -400.5 337.0 49.0 -419.0 -73.5 348.0 415.5 -123.0 -240.0 473.5 193.0 220.5 44.0 234.0 223.5 396.0 -471.5	207.00 279.00 -192.50 9.00 -123.00 -240.00 337.00 49.00

-117.5 130.5 -372.0 -277.5	
20 -260.5 78.0 -164.0 -422.5 -425.0 469.5 -214.5 481.5 463.0 126.0 318.5 -160.5 -371.0 -197.5 89.0 252.0 85.0 -337.5 -163.0 -482.0 -68.5 -205.5 -309.5 -441.5 51.0 -25.5 279.5 -279.0 -389.0 -128.5 -478.5 365.0 455.0 60.5 397.0 79.0 -417.5 -472.0 -326.5 -215.5	-478.50 365.00 -309.50 -441.50 85.00 -337.50 463.00 126.00

14 369.0 432.5 -43.5 -176.0 -436.5 -491.5 -133.0 -267.5 -336.0 -329.0 -433.0 -475.0 -290.0 230.5 -359.0 -177.0 -446.5 -435.5 51.0 283.0 177.0 161.0 107.5 -44.0 159.5 448.0 103.5 102.0	369.00 432.50 -290.00 230.50 -436.50 -491.50 -43.50 -176.00
18 -170.5 418.0 81.0 -50.5 -170.5 -407.0 -233.5 147.5 217.5 177.5 39.0 -493.0 -454.0 -471.0 314.0 25.5 -200.5 -169.5 -41.5 92.5 -313.5 488.0 201.5 229.0 -301.0 -334.0 -260.5 244.0 410.0 492.5 61.0 -432.0 396.0 223.0 130.5 57.0	410.00 492.50 -260.50 244.00 -301.00 -334.00 39.00 -493.00
7 250.0 -72.0 246.5 209.0 -325.0 139.0 206.0 -30.5 156.0 -93.5 223.0 458.5 439.5 376.0	0 -325.00 139.00 156.00 -93.50 206.00 -30.50 223.00 458.50 246.50 209.00 250.00 -72.00 439.50 376.00

Пример решения

```
using ll = long long;
using ull = unsigned long long;
using dl = double long;
#define all(x) x.begin(), x.end()
#include <vector>
#include <map>
#include <set>
#include <unordered_map>
#include <unordered_set>
#include <cmath>
#include <algorithm>
#include <iostream>
using namespace std;
#include <string>
```

```
int N = 4;
```

```
dl vec_product(dl ax, dl ay, dl bx, dl by, dl cx, dl cy) {
    return (ax - bx) * (cy - by) - (ay - by) * (cx - bx);
}
```

```
bool check_convex(vector<vector<dl>>& ps) {
    bool f1 = false;
    bool f2 = false;
    for (int i = 0; i < N; i++) {
        int p1 = i, p2 = (i + 1) % N, p3 = (i + 2) % N;
        dl vec_prod = vec_product(ps[p1][0], ps[p1][1], ps[p2][0], ps[p2][1], ps[p3][0], ps[p3][1]);

        if (vec_prod < 0) f1 = true;
        else if (vec_prod > 0) f2 = true;

        if (f1 && f2) return false;
    }
    return true;
}
```

```
int main() {
    int n;
    dl x, y;
    cin >> n;
    vector<vector<dl>> ps(n);
    for (int i = 0; i < n; i++) {
        cin >> x >> y;
        ps[i] = {x, y};
    }
```

```

}
int max_inside = -1;
vector<vector<dl>> ans;
for (vector<dl> p1 : ps) {
    for (vector<dl> p2 : ps) {
        if (p2[1] > p1[1] || (p2[1] == p1[1] && p2[0] < p1[0]) || (p2[0] == p1[0] && p2[1] ==
p1[1])) continue;
        for (vector<dl> p3 : ps) {
            if (p3[1] > p1[1] || (p3[1] == p1[1] && p3[0] < p1[0]) || (p3[0] == p1[0] && p3[1] ==
p1[1]) || (p3[0] == p2[0] && p3[1] == p2[1])) continue;
            for (vector<dl> p4 : ps) {
                if (p4[1] > p1[1] || (p4[1] == p1[1] && p4[0] < p1[0]) || (p4[0] == p1[0] && p4[1] ==
p1[1]) || (p4[0] == p3[0] && p4[1] == p3[1]) || (p4[0] == p2[0] && p4[1] == p2[1])) continue;
                vector<vector<dl>> mnogoug = {p1, p2, p3, p4};
                if (!check_convex(mnogoug)) continue;
                dl obxod = 0;
                for (int i = 0; i < N; i++) {
                    obxod += (mnogoug[(i + 1) % N][0] - mnogoug[i][0]) * (mnogoug[(i + 1) %
N][1] + mnogoug[i][1]);
                }

                if (obxod >= 0) continue;

                bool flag = true;
                int out_side = -1;
                set<pair<dl, dl>> outside_dots;
                for (int i = 0; i < N; i++) {
                    dl a = mnogoug[(i + 1) % N][1] - mnogoug[i][1];
                    dl b = mnogoug[i][0] - mnogoug[(i + 1) % N][0];
                    dl c = -a * mnogoug[i][0] - b * mnogoug[i][1];
                    int local_out_side = 0;
                    for (vector<dl> p : ps) {
                        if ((p[0] == p1[0] && p[1] == p1[1]) || (p[0] == p2[0] && p[1] == p2[1]) || (p[0]
== p3[0] && p[1] == p3[1]) || (p[0] == p4[0] && p[1] == p4[1])) {
                            outside_dots.insert({p[0], p[1]});
                            continue;
                        }
                        dl v = a * p[0] + b * p[1] + c;
                        if (v > 0) {
                            outside_dots.insert({ p[0], p[1] });
                            local_out_side++;
                        }
                        else if (v == 0) {
                            outside_dots.insert({ p[0], p[1] });
                        }
                    }
                }
                if (out_side == -1) {
                    out_side = local_out_side;

```

```

        continue;
    }
    else if (out_side != local_out_side) {
        flag = false;
        break;
    }
}
if (!flag) continue;

int in_side = n - outside_dots.size();

if (in_side > max_inside) {
    max_inside = in_side;
    ans = { p1, p2, p3, p4 };
}
}
}
}
}

cout << fixed;
cout.precision(2);

if (ans.size() == 0) {
    cout << 0 << endl;
    sort(all(ps), [](vector<dl>& p1, vector<dl>& p2) {return p1[0] < p2[0] || (p1[0] == p2[0] &&
p1[1] < p2[1]); });
    for (auto p : ps) cout << p[0] << " " << p[1] << endl;
}
else {
    for (auto p : ans) cout << p[0] << " " << p[1] << endl;
}
return 0;
}

```

Задача 6 (24 баллов) id 5146

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 по кольцевому маршруту, так как очень 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 искусстваами, 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 кольцевой 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*) кольцевого маршрута (начиная с вершины с меньшим номером, далее выбрать направление к вершине также с меньшим номером из двух возможных), 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) кольцевых маршрутов несколько, необходимо вывести самый безопасный.

Ввод	Вывод
------	-------

4 5 1 1 2 10 0.8 3 2 5 0.9 1 3 4 1.0 3 4 10 1.0 4 1 7 1.0 4	1 2 3 19 0.72
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	1 2 3 9 0.90

Примечание: гарантируется

- что искомый кольцевой 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
4 5 1 1 2 10 0.8 3 2 5 0.9 1 3 4 1.0 3 4 10 1.0 4 1 7 1.0 4	1 2 3 19 0.72
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	1 2 3 9 0.90
20 18 3 2 16 76 0.76 15 12 87 0.5 10 10 68 0.61 14 18 43 0.91 20 16 30 0.82 5 20 14 0.76 6 12 52 0.92 17 18 17 0.92 19 17 21 1.0 12 1 11 0.77 17 20 13 0.8 14 13 38 0.88 6 8 76 0.71 15 5 96 1.0 3 13 40 0.86 5 9 88 0.64 9 3 33 0.58 19 13 68 0.79 10 4	13 14 18 17 19 187 0.58

15	
13 10 1 7 9 33 0.95 1 1 28 0.97 12 4 30 0.76 12 7 48 0.9 4 1 75 0.94 2 9 18 0.93 2 12 19 0.78 2 5 30 0.85 2 7 33 1.0 5 5 5 0.84 5	2 9 7 12 118 0.62

28 28 4	1 11 3 23 161 0.51
8 24 96 0.98	
15 19 41 0.68	
8 15 87 0.7	
6 2 73 0.64	
21 15 6 0.68	
23 18 12 0.83	
12 20 14 0.78	
7 3 74 0.66	
27 16 90 0.8	
11 1 27 0.83	
28 11 67 0.63	
24 19 66 0.52	
2 17 1 0.6	
11 25 17 0.94	
3 11 63 0.91	
26 12 66 0.65	
18 15 44 0.76	
6 18 6 0.51	
15 4 85 0.72	
15 20 63 0.96	
11 2 2 0.89	
1 27 56 0.64	
10 1 22 0.76	
17 26 84 0.55	
16 26 14 0.67	
3 23 61 0.77	
16 6 69 0.69	
23 1 10 0.87	
20	
4	
15	
2	

29 29 12	4 8 5 12 280 0.50
6 25 10 0.52	
22 8 41 0.51	
9 7 7 0.94	
13 20 27 1.0	
18 23 36 0.81	
6 22 53 0.76	
12 5 82 0.85	
13 19 71 0.93	
21 22 41 0.62	
9 27 22 0.56	
1 21 64 0.64	
3 17 50 0.99	
8 4 71 0.97	
14 3 5 0.88	
25 1 28 0.96	
29 10 44 0.56	
3 2 37 1.0	
7 25 25 0.56	
21 20 13 0.64	
26 17 30 0.85	
7 7 23 0.64	
4 12 72 0.75	
8 5 55 0.81	
18 7 55 0.89	
25 26 44 0.55	
12 13 60 0.8	
3 4 47 0.92	
22 19 10 0.54	
29 14 77 0.93	
6	
21	
28	
16	
15	
9	
2	
11	
27	
17	
20	
1	

10 9 1 6 6 90 0.72 1 6 3 0.92 8 10 66 0.65 5 7 2 0.69 2 1 12 0.97 9 10 47 0.97 2 10 76 0.65 6 10 28 0.67 2 9 57 0.95 8	1 2 9 10 6 147 0.55
19 15 6 18 1 62 0.63 10 15 30 0.9 9 4 95 0.61 19 11 96 0.96 5 15 7 0.82 6 12 84 0.89 11 11 58 0.85 6 3 29 0.98 19 8 33 0.95 14 9 68 0.94 3 8 8 0.64 9 10 57 0.64 5 5 35 0.52 19 6 47 0.92 14 13 70 0.63 11 7 16 12 10 15	3 6 19 8 117 0.55

28 27 13	1 7 9 20 302 0.66
22 16 21 0.61	
8 25 20 0.96	
11 21 40 0.58	
6 24 60 0.96	
6 20 33 0.95	
6 22 80 0.68	
11 1 25 0.52	
12 21 96 0.64	
3 16 84 0.84	
16 9 20 0.91	
13 19 44 0.82	
9 20 84 0.87	
18 9 56 0.51	
1 7 100 0.97	
7 9 61 0.88	
5 7 37 0.91	
16 28 82 0.65	
19 24 65 0.84	
20 1 57 0.89	
27 15 53 0.64	
1 9 34 0.96	
11 3 26 0.55	
26 4 10 0.8	
15 14 29 0.98	
19 9 91 0.82	
14 14 31 0.71	
19 27 53 0.69	
12	
23	
19	
3	
16	
17	
4	
25	
24	
6	
28	
11	
10	

30 25 1	1 4 10 5 123 0.50
26 5 75 0.81	
6 13 3 0.83	
28 19 15 0.76	
8 14 77 0.52	
5 1 56 0.99	
5 10 41 0.64	
25 25 55 0.72	
21 1 16 0.71	
8 6 44 0.52	
7 7 36 0.55	
4 10 8 0.81	
4 27 84 0.78	
5 12 94 0.61	
20 22 71 0.98	
29 11 22 0.66	
3 19 95 0.67	
14 23 20 0.98	
27 24 9 0.9	
10 30 49 0.59	
28 26 84 0.81	
2 23 79 0.75	
1 4 18 0.98	
1 16 17 0.63	
8 28 76 0.52	
24 11 55 0.72	
6	

25 25 1	5 12 9 11 19 224 0.53
22 25 16 0.53	
12 17 30 0.82	
20 16 85 0.91	
11 20 7 0.63	
12 16 6 0.93	
7 18 44 0.79	
7 7 74 0.75	
21 18 37 0.79	
11 22 72 0.75	
5 24 46 0.75	
11 19 6 0.98	
4 24 59 1.0	
12 5 7 0.92	
7 11 18 0.65	
13 3 75 0.66	
9 12 75 0.73	
6 17 64 0.92	
9 11 36 0.81	
1 7 73 0.67	
19 5 100 0.99	
20 23 25 0.51	
14 15 77 0.62	
9 24 62 0.69	
23 13 97 0.51	
8 7 28 0.57	
2	

26 23 2	1 4 2 3 8 321 0.53
1 26 48 0.72	
17 15 16 0.86	
2 4 15 0.75	
12 17 36 0.85	
24 26 56 0.88	
9 14 80 0.77	
5 23 85 0.91	
20 3 36 0.67	
10 9 90 0.76	
16 11 87 0.77	
4 22 86 0.67	
10 23 81 0.68	
8 1 98 0.81	
14 12 25 0.62	
9 5 94 0.98	
6 3 44 0.74	
19 6 65 0.69	
2 3 20 0.97	
8 20 64 0.78	
3 8 91 0.95	
4 1 97 0.95	
22 22 2 0.68	
11 24 76 0.78	
13	
21	

29 23 3 25 5 44 0.54 7 8 26 0.78 27 9 21 0.69 23 2 62 0.92 1 21 95 0.6 19 20 23 0.86 1 16 56 0.91 19 27 25 0.77 9 14 5 0.88 8 12 15 0.94 23 1 53 0.81 24 9 81 0.95 15 17 93 0.59 2 17 6 0.88 24 7 71 1.0 21 9 91 0.6 13 27 11 0.81 14 8 79 0.97 23 14 43 0.96 29 10 46 0.85 20 5 22 0.65 13 17 93 0.54 28 4 40 0.93 3 4 29	7 8 14 9 24 262 0.63
17 15 4 6 3 15 0.64 13 11 28 0.75 9 2 55 0.62 12 1 95 0.93 3 4 85 0.85 11 4 12 0.73 6 11 56 0.94 7 2 87 0.83 6 14 29 0.68 3 12 10 0.94 8 6 40 0.81 11 9 75 0.84 7 12 43 0.64 16 11 88 0.56 6 12 1 0.93 9 2	3 4 11 6 12 164 0.51

5 14	
23 22 4 15 6 44 0.65 8 20 7 0.95 4 1 68 0.75 22 20 45 0.56 23 4 47 0.64 17 9 47 0.61 6 13 91 0.75 6 21 93 0.92 3 18 54 0.99 3 20 93 0.89 6 5 44 0.59 18 23 84 0.86 3 21 63 0.95 10 9 38 0.58 16 6 6 0.85 20 1 23 0.72 1 16 74 0.79 23 1 40 0.73 8 7 47 0.91 20 7 10 0.9 11 18 45 0.77 7 23 37 0.96 9 21 11 5	3 18 23 7 8 20 322 0.63

28 23 2 7 17 37 0.75 28 18 80 0.64 26 20 65 0.81 3 10 30 0.65 7 10 88 0.68 24 9 91 0.85 20 13 48 1.0 27 13 93 0.84 23 18 24 0.88 6 14 5 0.67 11 21 11 0.9 12 26 75 0.9 17 15 63 0.81 6 24 63 0.85 18 1 3 0.62 2 12 29 0.85 2 27 36 1.0 15 15 43 0.83 24 20 11 0.65 8 20 64 0.57 22 7 57 0.62 16 21 35 0.9 5 5 73 0.77 7 3	2 12 26 20 13 27 346 0.52
13 13 1 7 10 42 0.75 12 10 64 0.78 6 3 80 0.69 2 12 47 0.76 8 6 95 0.77 4 13 71 0.97 1 3 33 1.0 10 11 27 0.86 13 1 41 0.88 7 12 45 0.94 2 6 36 0.8 2 4 95 0.93 7 1 74 0.93 6	1 7 12 2 4 13 373 0.53

17 17 1 4 3 75 0.53 2 11 58 0.93 5 2 57 0.98 4 15 48 0.72 9 13 53 0.57 5 9 6 0.75 11 12 65 0.57 17 16 37 0.56 6 3 68 0.82 6 6 85 0.91 6 16 59 0.85 6 8 51 0.94 14 10 61 0.67 11 3 65 0.85 5 15 24 0.94 5 8 89 0.97 7 5 77 0.86 9	2 5 8 6 3 11 388 0.58
24 19 2 20 6 37 0.99 11 13 22 0.89 6 2 48 0.78 23 17 35 0.94 21 14 83 0.86 21 11 53 0.81 8 14 100 0.99 8 11 97 0.87 20 11 77 0.88 24 8 48 0.53 23 20 4 0.99 2 11 100 0.54 19 4 52 0.56 5 17 7 1.0 1 9 39 0.69 8 5 80 0.96 13 21 51 0.71 14 18 28 0.7 24 5 44 0.51 12 7	5 8 14 21 11 20 23 17 439 0.54

23 23 11	1 7 20 18 8 9 227 0.53
22 1 92 0.77	
6 18 14 0.69	
7 14 99 0.73	
17 3 10 0.6	
23 19 63 0.73	
8 18 7 0.82	
12 23 68 0.87	
22 18 57 0.87	
22 17 37 0.52	
21 7 97 0.65	
20 12 15 0.88	
20 13 3 0.68	
20 18 85 0.88	
20 22 14 0.64	
1 9 37 0.93	
23 11 24 0.9	
1 1 73 0.65	
18 15 60 0.89	
9 8 16 0.86	
11 15 18 0.64	
7 1 75 0.98	
9 10 81 0.71	
7 20 7 0.94	
6	
2	
11	
10	
5	
16	
22	
17	
3	
14	
19	

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 and b[0] != path[0]:
    continue
if b[0] == path[0] and len(path) < 3:
    continue

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

```

```

points = []
n, m, k = map(int, input().split())

```

```

edges = []
for i in range(m):
    edges.append(input())

```

```

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

```

```

d = dict()
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 pnt and s[1] not in pnt:
        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()

```

```

# print(d)

```

```

color = [0] * (n + 1)

```

```

for i in range(1, n + 1):

```

```
DFS(i, [i], 1, 0)
# print(*mxPath[0], mxPath[2], round(mxPath[1], 2))

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