TWGSS OI Team

2023/24 Team Formation Test

INFORMATION

Date	7 October 2023
Time	10:20 - 13:20 | 3 hours
Full Score	300

PROBLEMS

	Problem	*Limit*		*Score*	
		Time	*Memory*	*Subtasks*	*Total*
TF241	Hostel Score	1s	256MB	8 | 12 | 9 | 11	40
TF242	Green Walker	1s	256MB	8 | 10 | 22	40
TF243	Ratism Team	1s	256MB	7 | 6 | 14 | 17 | 16	60
TF244	Problematic Username	1s	256MB	17 | 15 | 16 | 13 | 9	70
TF245	King's Challenge	1s	256MB	14 | 22 | 29 | 25	90

REMINDERS

- All problems are divided into subtasks. All test cases in a subtask must be passed to get points.
- You may attempt the problems using C++20 or Python 3 (secondary language). It is not guaranteed that the problems are solvable using secondary language.
- Unless otherwise specified, your output will be automatically fixed as follows:
  - Trailing spaces in each line will be removed.
  - An end-of-line character will be added to the end of the output if not present.
- 64-bit integers may be required in some problems, use `long long` in C++ if needed.
- All stories are hypothetically written. In case of coincidences, they are coincidences.

TF241 Hostel Score

Score	Limit	
40	1s	256MB

Time files, it has already been a year after Daniel had the memorable journey on an exciting Thursday. The change from secondary school to the university is tremendous. In the new environment of The Coder's University of Hong Kong (CUHK), the way forward is full of uncertain and discoveries…

Although Tsuen Wan has now arisen, it still takes a lot of time travelling from Tsuen Wan to CUHK. Therefore, to save his precious time doing "five things in university", he decides to register a hostel!

As university is an epitome of the society, housing problem is critical in CUHK. There is not enough hostel for all students. Therefore, whether or not they get a hostel depends on their **hostel scores**, which depends on three factors: **District**, **Housing** and **Extra-Curricular Activities**.

District (Max. 45)

Travelling Time	*Score*
90 minutes or more	43
88 - 89 minutes	42
86 - 87 minutes	41
84 - 85 minutes	40
⋮	⋮
12 - 13 minutes	4
11 minutes or less	0

Number of Interchanges	*Score*
2 or more	2
1	1
0	0

Housing (Max. 35)

Age of Building	*Score*
50 years or more	7
40 - 49 years	5
30 - 39 years	3
29 years or less	0

Extra	*Score*
Forth-floor or above without lift	3

Floor Area per Capita (rounded off to nearest ft²)	*Score*
151 ft² or more	0
131 - 150 ft²	1
111 - 130 ft²	3
91 - 110 ft²	5
71 - 90 ft²	8
51 - 70 ft²	12
31 - 50 ft²	20
30 ft² or less	25

Extra-Curricular Activities (Max. 20)

Item	*Role*	*Score*
Student Union	Committee	20
Orientation Camp	President	9
	Member	5
Singing Contest	President	8
	Member	3
Happy Run	Participant	2
Walk for Green	Participant	2

The score of each factor is bounded by a maximum value. The total **hostel score** is calculated by adding up the sum of the three factors. The calculation is way too complicated for the IOI medallist to handle! In order to estimate the chance of getting a hostel place, Daniel decides to ask you, a bright programming star, to write a program finding the hostel score for him!

INPUT

The first line consists of two integers, T and I, separated by a space, representing the **travelling time** and the **number of interchanges** needed travelling from Daniel's house to CUHK respectively.

The second line consists of a string, which is either `without lift` or `with lift`, describing whether there is a lift in Daniel's building.

The third line consists of four integers, Y, F, A and C, each separated by a space, representing the **age of building**, the **floor number**, the **floor area** and the **number of capital (people)** living in Daniel's flat respectively.

The fourth line consists of a string S, describing the **extra-curricular activities** that he takes part in. Activity are in the format of `[Item] [Role]`, and each is separated by `, `.

OUTPUT

Output an integer on the first and only line, the **hostel score** that Daniel has.

SAMPLE TESTS

Input	*Output*
41 2 without lift 56 4 152 3 Walk for Green Participant, Orientation Camp Member	49

The **hostel score** can be calculated as follows:

- **District**: Travelling Time + Interchange $= 18 + 2 = 20$
- **Housing**: Age of Building + Extra + Floor Area per Capita $= 7 + 3 + 12 = 22$
- **Extra-Curricular Activities**: Walk for Green Participant + Orientation Camp Member $= 2 + 5 = 7$

Final **hostel score** $= 20 + 22 + 7 = 49$.

2

```
90 5
with lift
10 56 1000 1
Student Union Committee, Happy Run Participant
```

```
65
```

The **hostel score** can be calculated as follows:

- **District**: Travelling Time + Interchange $= 43 + 2 = 45$
- **Housing**: Age of Building + Extra + Floor Area per Capita $= 0 + 0 + 0 = 0$
- **Extra-Curricular Activities**: Student Union Committee + Happy Run Participant $= 20 + 2 = 22$, bounded by the maximum of 20

Final **hostel score** $= 45 + 0 + 20 = 65$.

CONSTRAINTS

$0 \leq T \leq 240$

$0 \leq I \leq 5$

$0 \leq Y \leq 200$

$0 \leq F \leq 100$

$0 \leq A \leq 1000$

$1 \leq C \leq 10$

SUBTASKS

	Points	Constraints
1	8	$50 \leq Y \leq 200$ $0 \leq F \leq 3$ $\frac{A}{C} \geq 151$ S is empty
2	12	S is empty
3	9	S contains at most 1 activity
4	11	No additional constraints

TF242 Green Walker

Score	Limit
40	1s | 256MB

Walking in the CUHK campus, Daniel sees some students wearing the *"Go Green! Be Sunny!"* t-shirt. He suddenly realises that he can gain more hostel points by participating the "Walk for Green" activity!

There is a path in the university that there are N **card machines** placed all the way through, numbered 1 to N. The distance between each adjacent **card machine** is the same. To obtain points in "Walk for Green", Daniel has to tap the card on each and every **card machines** once in a single tour, with a slight twist – each walk between two **card machines** must be in different durations! As a good student that strictly follows the rule of academic honesty, Daniel decided to not stand still after each walk to wait until a different duration is achieved. Instead, he decided to ask you, a bright programming star, to write a program constructing a sequence of visiting **card machines** such that he can obtain points in a single tour by walking in a constant speed!

INPUT

The first and only line consists of an integer N, the number of **card machines**.

OUTPUT

Output N integers. The i^{th} integer should be i^{th} the **card machine** to be visited in the tour.

Note that all elements should be distinct, and the absolute differences of adjacent elements should also be distinct.

If there are multiple valid sequences, output any one of them.

SAMPLE TESTS

	Input	Output
1	5	5 1 4 2 3

- The distance from machine 5 to machine 1 is $|5-1|=4$.
- The distance from machine 1 to machine 4 is $|1-4|=3$.
- The distance from machine 4 to machine 2 is $|4-2|=2$.
- The distance from machine 2 to machine 3 is $|2-3|=1$.

As all absolute distance differences of adjacent machines are different, this is a valid solution.

CONSTRAINTS

$1 \leq N \leq 2 \times 10^5$

SUBTASKS

	Points	Constraints
1	8	$1 \leq N \leq 4$
2	10	$1 \leq N \leq 8$
3	22	No additional constraints

TF243 Ratism Team

Score	Limit	
60	1s	256MB

As The Coder's University, the school takes part in the International Collegiate Programming Contest (ICPC) and got great results in past years! In this team contest, each team consists of three teammates. During the Art Fair at The University Mall, a live programming performance was held, attracting many students to apply for joining the CUHK ICPC Team! Therefore, a Team Formation Test (TFT) would be held for ranking the members and arranging teams.

Joining the code forces of CUHK, Daniel receives a message from the coach, *"Hey Daniel! You are so famous in HKOI! Don't worry, you are very safe getting the seat for World Finals! Please help us to write a program for arranging teams if you are free!"* Little does the coach know, Daniel is busy *Walking for Green*! Therefore, Daniel asks you, a bright programming star, to help him with that!

Of course, everyone wish to group with strong programmers, so that they can win without even touching the keyboard! To resolve this, the coach has adopted a "ratism" set of rules:

- Every member is allowed to specify zero, one or two other member(s) who he / she wants to team with.
- The rank of each member is determined in the TFT. No two members have the same rank.
- The highest-ranked member who is not yet in any team can create his / her own team. He / she would choose all his / her teammate preferences that is not yet in a team them to join his team immediately. Then, if the team still do not have three teammates, he / she will select the next highest-rank member, and if needed, the second-next highest-rank member, as his teammate.
- The above process is repeated, until all members are in some teams.

There is a total $3N$ members to be split into N teams of 3. Given the preferences of all members and the TFT ranking, you are going to write a program determining the teams formed. Note that in order to convey the "ratism" principle, the team list should be outputted in ascending order of TFT rankings (from strongest to weakest) of the strongest member within the team. In each team, you should also output the teammates in ascending order of their TFT rankings.

INPUT

The first line consists of an integer N, the number of teams to be formed.

The i^{th} line of the next $3N$ lines describes the preference of the i^{th} member. The line starts with a string s_i, the name of the member. Followed by an integer m_i, the number of other members that he / she wants to team with, in which m_i more integers follow, $x_{i,1}, \ldots, x_{i,m_i}$, refering to the indexes of the other members that he / she wants to team with.

The last line consists of $3N$ integers, $R_1, R_2, \ldots, R_{3N}$, which R_i is the index of the member who gets the i^{th} rank in TFT.

OUTPUT

Output N lines, the teams formed in ascending order of TFT rankings of their strongest member.

The i^{th} line should describe the i^{th} team, being the names of the teammates in ascending order of their TFT rankings, each separated by a space.

SAMPLE TESTS

1

Input	Output
2 ada 1 4 bob 2 1 4 chris 1 5 donna 2 1 2 eve 0 frank 2 3 5 3 1 2 4 5 6	chris ada eve bob donna frank

- The highest-ranked member is `chris` (member 3). Therefore, `chris` can create a new team. As he invited `eve` (member 5) to join his team while `eve` is not in any team yet, `eve` joins `chris`' team. Then, the next highest ranked member, `ada` (member 1) also joins `chris`' team.
- The next highest-ranked member who is not in any team yet is `bob` (member 2). Therefore, `bob` can create a new team. He wishes to team with `ada` (member 1), but since `ada` is already in team 1, `ada` cannot join `bob`'s team. Then, as he also wants to team with `donna` (member 4), and `donna` wasn't in any team yet, `donna` joins `bob`'s team. Finally, the next highest-ranked member not yet in any team, `frank` (member 6), joins `bob`'s team.

Therefore, team 1 consists of `chris`, `eve` and `ada`, and team 2 consists of `bob`, `donna` and `frank`. Please note that you have to output the names within a team in ascending order of their TFT rankings.

2

2 ada 1 4 bob 2 1 4 chris 1 5 donna 2 1 2 eve 0 frank 2 3 5 1 2 3 4 5 6	ada bob donna chris eve frank

CONSTRAINTS

$1 \leq N \leq 10^5$

s_i consist of lowercase English letters (`a` - `z`)

$1 \leq \text{length of } s_i \leq 10$

$s_i \neq s_j$ for $i \neq j$

$0 \leq m_i \leq 2$

$1 \leq x_{i,j}, p_i \leq 3N$

$x_{i,j} \neq i$

$x_{i,j_1} \neq x_{i,j_2}$ for $j_1 \neq j_2$

$R_i \neq R_j$ for $i \neq j$

SUBTASKS

	Points	Constraints
1	7	$N = 1$
2	6	$R_i = i$ $m_i = 0$
3	14	$R_i = i$
4	17	$1 \leq N \leq 1000$
5	16	No additional constraints

TF244 Problematic Username

Score	Limit	
70	1s	256MB

"The ICPC TFT is much more fun than the TFT in secondary school! Luckily I am now a university student—"

*"TWGSS Online Judge - A new account is registered: 'f***'"* A Discord bot message pops up on Daniel's phone.

"What the username?!?!?" Daniel quickly decides that he has to impose restrictions to the usernames on the Judge as his one of the hundreds last things contributing to his secondary school team.

There are N accounts in the Judge. By closer inspection, Daniel finds out that this is not the only problematic username. Therefore, he composes a list of M **inappropriate words** that should not appear in usernames, and some other rules that he wants to impose. As his friend Jack has already done the *Problem Descriptor*, Daniel gives some notes to him and asks him to write a program that can automatically fix all usernames on the Judge. As Jack is busy, he decides to ask you, a bright programming star, for help instead *(free labour!)*.

Permitted characters for usernames include:

- English letters (`A`-`Z`, `a`-`z`)
- Numbers (`0`-`9`)
- Underscore (`_`)
- Full stop (`.`)

Procedure of **problematic usernames correction**:

1. Replace all characters of **inappropriate words** by **underscores (`_`)**.
   - The check of inappropriate words is **case-insensitive** (uppercase and lowercase of an English alphabet is considered the same).
   - In case if two inappropriate words **overlapped**, the entirety of the two inappropriate words should be replaced.
2. Replace all **spaces (` `)** by **underscores (`_`)**.
3. Replace all **commas (`,`)** by **full stops (`.`)**.
4. If there are **consecutive** (two or more next to each other) **full stops (`.`)**, change them to a **single full stop (`.`)**.
5. Delete all **non-permitted characters**.
6. If $\text{length of username} < 3$, append **underscores (`_`)** to the end of the username until $\text{length of username} = 3$.
7. If $\text{length of username} > 12$, delete all **full stops (`.`)**.
8. If $\text{length of username} > 12$, delete all **underscores (`_`)**.
9. If $\text{length of username} > 12$, delete the last character until $\text{length of username} = 12$.

Note that the procedure should be performed one by one following the order.

After correcting all problematic usernames, a **distinctness check** will be perform to all usernames. For identical usernames, an **asterisk (`*`)** should be added in front of all appearance of them, such that Daniel can easily notice these cases and handle them manually. Note that the check for identical usernames is **case-insensitive** (uppercase and lowercase of an English alphabet is considered the same).

INPUT

The first line consists of two integers, N and M, separated by a space, representing the number of accounts and the number of inappropriate words respectively.

The i^{th} line of the next N lines contains a string S_i, representing the username of the i^{th} account. Note that S_i may contains spaces.

The i^{th} line of the next M lines contains a string F_i, representing the i^{th} inappropriate words.

OUTPUT

Output N lines.

On the i^{th} line, output the resultant username of the i^{th} account after the **problematic usernames correction** and the **distinctness check**.

SAMPLE TESTS

Input	Output
5 3 HKOIOI So fun oi >_< ...IT._.Helper ...IT._.Manager IOIOI_ hkoi noi ioi	*_____ So_fun_oi__ .IT._.Helper IT_Manager *_____

CONSTRAINTS

$1 \leq N \leq 10^5$

$1 \leq M \leq 100$

$1 \leq \text{length of } S_i, \text{length of } F_i \leq 20$

S_i consists of printable ASCII characters

F_i consists of lowercase English letters (`a` - `z`)

SUBTASKS

	Points	Constraints
1	17	S_i only consists of lowercase English letters (`a` - `z`) $3 \leq \text{length of } S_i \leq 12$ Each S_i contains at most 1 inappropriate word There is no identical usernames after the **problematic usernames correction**
2	15	Each S_i contains at most 1 inappropriate word There is no identical usernames after the **problematic usernames correction**
3	16	$M = 0$
4	13	S_i only consists of lowercase English letters (`a` - `z`)
5	9	No additional constraints

TF245 King's Challenge

Score	Limit	
90	1s	256MB

After discovering the inappropriate usernames in the Judge, Daniel understands the importance of being polite! Therefore, despite he has zero interest in the topic, he still attends the "General Education" lecture, to respect the lecturer and his marks! This time, the lecture is about traditions. As Daniel recognises sleeping is the only tradition, he decides to directly fall asleep! As he is feeling spiritually "interested", he suddenly seemingly hears a word that catches his ear – Light of The Cross (LTC)! What does that come from? He wonders!

A vivid situation suddenly comes up in front of Daniel – King Charles is receiving a challenge – Spiritual Boundary Assessment (SBA) from the Ancient God, Haridra Lord Teandrios (HLT)! To test the limit of King Charles' spirit, HLT gives King Charles a square land of size $N \times N$. Each cell originally has some level of fertility. For each moment, HLT would perform one of the following operations:

Type	Format	Operation
LTC	r c	LTC is used in cell (r, c) such that the fertility of all cells that have a positive value in row r and column c is reduced by 1.
Inspect	d x	If $d =$ `R`, the total fertility of all cells in the x^{th} row is queried. If $d =$ `C`, the total fertility of all cells in the x^{th} column is queried.

The challenge would end once the total fertility of all cells in any row, column or main diagonal is zero.

"Wake up! The lecture is about to end!" Daniel is called by the person seating next to him.
"Such a rewarding lecture!" Daniel said.

Daniel quickly decides to write this wonderful story down and ask you, a bright programming star, to write a program for King Charles to overcome the challenge by HLT!

INPUT

The first line consists of two integers, N and Q, separated by a space, representing the side length of the square land and the number of operations respectively.

The next N line each consists of N integers, each separated by a space. The j^{th} character of the i^{th} line represents the fertility $F_{i,j}$ of the cell (i, j).

The i^{th} line of the next Q lines consists of an integer t_i and an operation. If $t_i =$ `1`, a **"LTC"** operation follows. If $t_i =$ `2`, an **"Inspect"** operation follows.

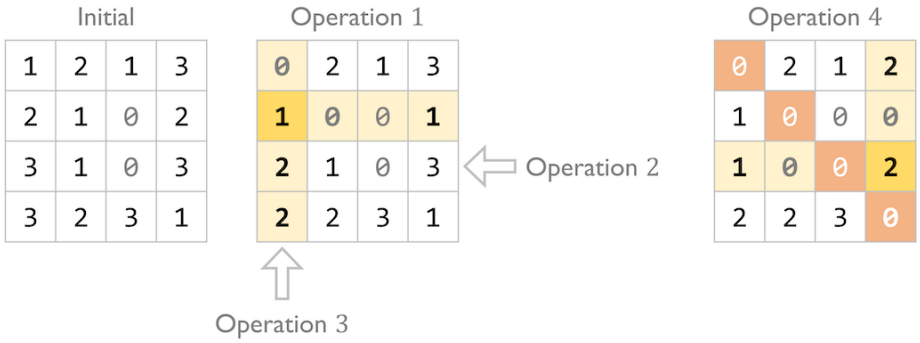
OUTPUT

For each **"LTC"** operation, output `God Save the King!` if the total fertility of all cells in any row, column or main diagonal is zero and **terminates the program immediately.** Otherwise, output `Keep Going!`.

For each **"Inspect"** operation, output the answer of the query.

SAMPLE TESTS

Input	Output
4 5 1 2 1 3 2 1 0 2 3 1 0 3 3 2 3 1 1 2 1 2 R 3 2 C 1 1 3 4 2 R 1	Keep Going! 6 5 God Save the King!



- After operation 1, total fertility of all rows, columns and main diagonals are greater than zero. Therefore, `Keep Going!` is outputted.
- For operation 2, sum of row $3 = 2 + 1 + 0 + 3 = 6$.
- For operation 3, sum of column $1 = 0 + 1 + 2 + 2 = 5$.
- After operation 4, total fertility of the top-left bottom-right diagonal $= 0 + 0 + 0 + 0 = 0$. Therefore, `God Save the King!` is outputted and the program is terminated.
- As the program is terminated, operation 5 is ignored.

CONSTRAINTS

$1 \leq N, Q, F_{i,j} \leq 1000$

$t_i =$ `1` or `2`

$1 \leq r, c, x \leq N$

$d =$ `R` or `C`

SUBTASKS

	Points	Constraints
1	14	$N = 1$
2	22	$1 \leq N \leq 100$
3	29	$T_i = 2$
4	25	No additional constraints