Problem A. Ascending Photo

Source file name:	photo.c, photo.cpp, photo.java, photo.py
Input:	`Standard`
Output:	`Standard`

The members of the No-Weather-too-Extreme Recreational Climbing society completed their 100th successful summit today! To commemorate the occasion, we took a picture of all the members standing together in one row, to use for marketing purposes.

However, the photograph looks messy; as usual, the members refused to order themselves in any kind of aesthetically pleasing way. We will need to reorder the picture.

Figure A.1: **This picture** has been cut up and pasted back together to solve Example Input 1.

Our research tells us that having the climbers in ascending (non-decreasing) height order from left to right will be most visually appealing. We must cut up the picture we have and somehow paste it back together in this order.

Find the minimum number of cuts you need to make to put the photograph into ascending order.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 10^6$), the number of people in the photo.
- One line with n integers $h_1, ..., h_n$ ($1 \leq h_i \leq 2 \cdot 10^9$ for each i), the heights of the people in the photograph, from left to right.

Output

Output the minimum number of cuts needed to rearrange the photograph into any one ascending (non-decreasing) height order from left to right.

Example

Input	Output
11 3 6 12 7 7 7 7 8 10 5 5	4
3 5000000 5500000 7000000	0
12 1 2 2 3 3 1 2 3 4 1 2 3	6

Problem B. Boss Battle

Source file name:	boss.c, boss.cpp, boss.java, boss.py
Input:	`Standard`
Output:	`Standard`

You are stuck at a boss level of your favourite video game. The boss battle happens in a circular room with n indestructible pillars arranged evenly around the room. The boss hides behind an unknown pillar. Then the two of you proceed in turns.

- First, in your turn, you can throw a bomb past one of the pillars. The bomb will defeat the boss if it is behind that pillar, or either of the adjacent pillars.
- Next, if the boss was not defeated, it may either stay where it is, or use its turn to move to a pillar that is adjacent to its current position. With the smoke of the explosion you cannot see this movement.

The last time you tried to beat the boss you failed because you ran out of bombs. This time you want to gather enough bombs to make sure that whatever the boss does you will be able to beat it. What is the minimum number of bombs you need in order to defeat the boss in the worst case? See Figure **B.1** for an example.

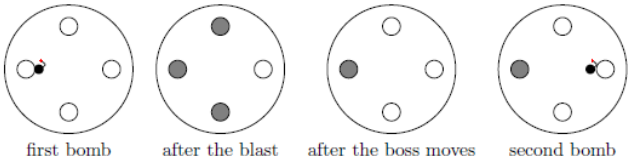

Figure **B.1**: Example for $n = 4$. In this case 2 bombs are enough. Grey pillars represent pillars where the boss cannot be hiding. The bomb is represented in black.

Input

The input consists of:

- One line with a single integer n ($1 \leq n \leq 100$), the number of pillars in the room.

Output

Output the minimum number of bombs needed to defeat the boss in the worst case.

Example

Input	Output
4	2
7	5

Problem C. Connect the Dots

Source file name:	dots.c, dots.cpp, dots.java, dots.py
Input:	`Standard`
Output:	`Standard`

A famous logical problem is that of connecting 9 dots on a paper by drawing 4 line segments with a pencil, while never lifting the pencil from the paper. While this is easy enough (although it requires some thinking outside of the box), Simone has recently been building a game called "Connect the Dots" around a generalisation of the concept.

In Connect the Dots, you are presented with a 4×4 regular grid of dots. Each dot is given a unique number between 1 and 16. The task is then to connect the dots in order by their numbers, starting with dot 1 and ending with dot 16. The dots should be connected using *as few line segments as possible*, starting at dot 1, with the end of each segment being the start point of the next. The segments are allowed to intersect and overlap one another. Additionally, it is allowed to pass through other points while trying to connect the current point. This means, for example, that visiting the first four points in the order $1, 4, 2, 3, 2, 4, ...$ is acceptable. Formally, the sequence $1, 2, ..., 15, 16$ must be a subsequence of the sequence of dots visited.

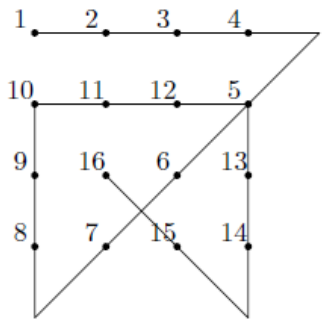

Figure C.1: A solution to the first sample.

Simone asked you to try the puzzle out, while betting you a balloon that it would be too hard. Prove her wrong by writing a program that solves the puzzle for you!

Input

The input consists of:

- 4 lines, each with 4 integers, the numbers of the dots in the grid. The jth number on the ith line is the number of the jth dot in the ith row of the grid of dots.

The 16 numbers in the input are all between 1 and 16 (inclusive) and pairwise distinct.

Output

Output the minimum number of line segments needed to connect all the dots in order.

Example

Input	Output
1 2 3 4 10 11 12 5 9 16 6 13 8 7 15 14	6
1 2 3 4 8 9 10 11 7 15 16 12 6 14 13 5	7

Problem D. Dunglish

Source file name:	dunglish.c, dunglish.cpp, dunglish.java, dunglish.py
Input:	`Standard`
Output:	`Standard`

A group of confused Dutchmen. Public domain.

A confused Dutchman trying to speak English could say "*I am in the war*", even though there is no hostile activity going on. The confusion here is that the English sentence "*I am confused*" is translated in Dutch as "*Ik ben in de war*", which is phonetically ("sounding") quite close to the first sentence. Such confusion leads to much enjoyment, but can complicate matters a bit.

Given a sentence in Dutch and a dictionary containing both correct translations as well as phonetic (incorrect) translations of individual words, find the translation of the sentence and indicate whether it is correct, or in case there is more than one find the total number of correct and incorrect translations. A sentence is correctly translated when each word of the sentence is correctly translated.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 20$), the number of words in the Dutch sentence.
- One line with n words, the Dutch sentence s
- One line with an integer m ($1 \leq m \leq 10^5$), the number of words in the dictionary.
- m lines, each with three strings d, e and c, a Dutch word, the English translation, and "`correct`" if this is the correct translation or "`incorrect`" otherwise.

A word consists of between 1 and 20 lowercase letters. Each word in s appears at least once as a Dutch word in the dictionary, no word appears more than 8 times as a Dutch word in the dictionary, and each combination of a Dutch and English word appears at most once.

Output

In case there is only a single translation of s, output one line with the translation followed by one line with "`correct`" or "`incorrect`". In case there is more than one translation, output one line with the number of possible correct translations followed by "`correct`", and one line with the number of possible incorrect translations followed by "`incorrect`".

Example

Input
7 als mollen mollen mollen mollen mollen mollen 4 als when correct mollen moles correct mollen destroy correct mollen mills incorrect
Output
64 correct 665 incorrect

Input
5 de zuigers zijn buiten werking 6 zijn are correct banaan banana correct de the correct zuigers suckers incorrect buiten out correct werking working incorrect
Output
the suckers are out working incorrect

Problem E. English Restaurant

Source file name:	restaurant.c, restaurant.cpp, restaurant.java, restaurant.py
Input:	`Standard`
Output:	`Standard`

After finding a suitable residence in the Swiss Alps, Robin thought he finally had what it takes to be happy. But every day he woke up feeling that something was missing. Maybe it was the lack of English food in these parts? Spotting a market opportunity and solving his problem at the same time, he teamed up with the famous English chef Jim to open a restaurant nearby. While he has no doubts about Jim's talents as a chef, Robin wants to be sure that opening an English restaurant in the Swiss Alps is a good idea.

Fortunately, as a local, he knows the habits of restaurant customers. At the stroke of each hour, a group of people arrives of size that is uniformly random between 1 and g people (inclusive). The group finds the smallest completely unoccupied table that fits the group and occupies it. If they can not find such a table, the group leaves with great disappointment. Once seated, a group never leaves until the restaurant closes, as Jim has no difficulty keeping the guests entertained.

As an example, suppose the restaurant has 3 tables of capacities 5, 8 and 9. If groups of sizes 5, 10 and 3 arrive (in that order), in the end there will be 8 people in the restaurant. The first group occupies the table of capacity 5, the second group leaves and the last group occupies the table of capacity 8.

Robin plans to keep his restaurant open for t hours in total. In the restaurant business the most important metric is the expected number of people in the restaurant when it is closes. Can you help Robin calculate the expected occupancy after t hours?

Input

The input consists of:

- One line with three integers n, g, t ($1 \leq n \leq 100, 1 \leq g \leq 200, 1 \leq t \leq 100$), the number of tables in the restaurant, the maximum group size, and the number of hours the restaurant is open.
- One line with n integers $c_1, ..., c_n$ ($1 \leq c_i \leq 200$ for each i) giving the capacities of the tables.

Output

Output the expected number of people in the restaurant when it closes. Your answer should have an absolute or relative error of at most 10^{-6}

Example

Input	Output
3 3 2 1 2 3	3.666666667
4 11 4 10 10 10 10	20.000000000
4 3 3 4 1 3 2	5.888888888888

Problem F. Automaton

Source file name:	automaton.c, automaton.cpp, automaton.java, automaton.py
Input:	`Standard`
Output:	`Standard`

Pepito Pérez found the next definition in **Wikipedia**: a **Nondeterministic Finite Automaton (NFA)** is represented formally by a 5-tuple, $M = (Q,\ \Sigma,\ q_0,\ F,\ \Delta)$, consisting of:

- a finite set of states Q,
- a finite set of input symbols (Alphabet) Σ,
- an initial (or start) state $q_0 \in Q$,
- a set of states F distinguished as accepting (or final) states, $F \subseteq Q$
- a transition function $\Delta : Q \times \Sigma \to P(Q)$

Here, $P(Q)$ denotes the power set of Q. Let $w = a_1 a_2 \ldots a_n$ be a word over the alphabet Σ. The Nondeterministic Finite Automaton M accepts the word w if a sequence of states, r_0, r_1, ..., r_n, exists in Q with the following conditions:

1. $r_0 = q_0$
2. $r_{i+1} \in \Delta(r_i, a_{i+1})$, for $i = 0, 1, \ldots, n-1$
3. $r_n \in F$

In words, the first condition says that the machine starts in the start state q_0. The second condition says that given each character of string w, the machine will transition from state to state according to the transition function Δ. The last condition says that the machine accepts w if the last input of w makes the machine to be in one of the accepting states. In order for w being accepted by M it is not required that every state sequence ends in an accepting state, it is sufficient if one does. Otherwise, i.e. if it is impossible at all to get from q_0 to a state from F by following w, it is said that the automaton rejects the string. The set of strings that M accepts is the language recognized by M and this language is denoted by $L(M)$.

Pepito implemented the nondeterministic finite automaton but the execution time is too big for word recognition. *Pepito* has heard that every word could be recognized in $O(|w|)$, but *Pepito* has no idea on how to do it!, Can you help him?

Input

Input begins with an integer t ($1 \leq t \leq 10$), the number of test cases. The first line of the test case contains four space-separated positive integers n m s k ($1 \leq n \leq 18$, $1 \leq m \leq 8424$, $1 \leq s \leq 26$, $1 \leq k \leq n$), which represent, the number of states, the number of transitions, the number of alphabet symbols and the number of states of acceptance, respectively. The second line contains exactly s space-separated different symbols σ_1, σ_2, σ_3, ..., σ_s ($\sigma_i \in \{a,\ b,\ c,\ d,\ \ldots,\ z\}$, σ_i is any lowercase letter of the English alphabet). The alphabet Σ is sorted lexicographically. The third line contains exactly k space-separated nonnegative integer numbers F_1, F_2, ..., F_k ($0 \leq F_i < n$, for $0 \leq i \leq k$), these are the acceptation states. The test case continues with m lines each of them containing a transition of the automaton x y c ($0 \leq x, y < n$, $c \in \Sigma$), where x y and c are the origin, destination, and label of the transition, respectively. Then a line is presented that contains a positive integer q that represents the total of words that are going to be evaluated in the automata, finally, q lines are presented, each one containing a word w ($w \in \Sigma^+$, $1 \leq |w| \leq 10^5$, that is, the words w have a length between 1 and 10^5)

Output

For each test case, you should print q single lines, each line containing the word `Yes` or `No` depending if the word $w \in L(M)$.

Example

Input	Output
1 5 12 2 2 a b 0 2 0 3 a 0 1 b 1 1 a 1 1 b 1 3 b 2 0 a 2 1 a 2 4 b 3 2 a 3 4 b 4 2 a 4 4 a 10 aaaaaaaa abababab bbbbaaa a b abaaba bbaab babab bbbaaba bbaabbaa	Yes No Yes No No Yes No No Yes Yes

Use fast I/O methods

Problem G. Warships

Source file name:	warships.c, warships.cpp, warships.java, warships.py
Input:	`Standard`
Output:	`Standard`

Tobby and **Naebbirac** are playing one version of the game **Warships** in which each player has a fleet of ships to fight against pirates and other players. In this game, there are only three types of ships: Battleships, Destroyers and Cruisers. **Tobby** decided to become strong rapidly and, for that reason, he just started constructing all ships that he can of a single type. Curiously, **Naebbirac** has the same idea and he is doing exactly the same kind of fleet (probably with another type of ship). No player could have more than a million of ships.

Due specific characteristics, attack power and speed of each ship's type the game use a relation of superiority between each pair of ship's type to predict the result of each battle. By instance, a number B of Battleships will always win against a number D of Destroyers, unless D is twice or more the value of B. Similarly, a number D of Destroyers will always win against C Cruisers, unless C is twice or more the value of D. Finally, a number C of Cruisers will always win against B Battleships, unless B is twice or more the value of C. Also, in case of a battle using the same type of ships the winner will be the player who fights with the greater amount of ships; if both players have equal amount of ships then there is no winner because strength is the same (is a draw).

Today is Saturday, and as every week is coming a War Time event. Now, they want two know who is stronger!!! They want to know who will win a battle if both players use their entire fleet on that. Can you help to determine the winner?

Input

The input consists of several test cases, read until the end of file (EOF). Each test case consists of exactly two lines. The first line contains the fleet description of **Tobby**, while the second line contains the fleet description of **Naebbirac**. The description of a fleet will be composed by two space-separated elements; the ship's type and a positive integer X ($1 \leq X \leq 10^6$) denoting the number of ships in the fleet respectively.

Output

For each test case, your program must print a line with the name of the winner, or print a line with the word "Draw" without quotes if there is no winner.

Example

Input	Output
Cruisers 150 Battleships 200 Destroyers 150 Cruisers 350 Destroyers 150 Destroyers 150	Tobby Naebbirac Draw

Use fast I/O methods

Problem H. High Score

Source file name:	score.c, score.cpp, score.java, score.py
Input:	`Standard`
Output:	`Standard`

The Great Pyramid of Giza; one of the seven wonders. By Nina-no, licensed under CC BY-SA 3.0

Mårten and Simon enjoy playing the popular board game Seven Wonders, and have just finished a match. It is now time to tally the scores.

One of the ways to score in Seven Wonders is through the use of *Science*. During the game, the players may collect a number of Science tokens of three different types: *Cog*, *Tablet*, and *Compass*. If a player has a Cogs, b Tablets and c Compasses, that player gets $a^2 + b^2 + c^2 + 7 \cdot min(a, b, c)$ points.

However, the scoring is complicated by the concept of *Wildcard Science* tokens. For each Wildcard Science token a player has, she may count that as one of the three ordinary types of Science tokens. For instance, the first player in Sample Input 1 has 2 Cogs, 1 Tablet, 2 Compasses, and 1 Wildcard Science, so could thus choose to have the distributions (3, 1, 2), (2, 2, 2) or (2, 1, 3) of Cogs, Tablets and Compasses, respectively. The possible scores for this player are then $3^2 + 1^2 + 2^2 + 7 \cdot 1 = 21$, $2^2 + 2^2 + 2^2 + 7 \cdot 2 = 26$ and $2^2 + 1^2 + 3^2 + 7 \cdot 1 = 21$ depending on how the Wildcard Science is assigned. Thus, the maximum score for this player is 26.

Given the number of tokens each player in the game has, compute the maximum possible score that each of them can achieve if they assign their Wildcard Science tokens optimally.

Input

The input consists of:

- One line with an integer n ($3 \leq n \leq 7$), the number of players in the game.
- n lines, each with four integers a, b, c, d ($0 \leq a, b, c, d \leq 10^9$), giving the number of Cog, Tablet, Compass, and Wildcard Science tokens of a player.

Output

For each player, in the same order they are given in the input, output the maximum score the player may get.

Example

Input	Output
3 2 1 2 1 3 2 1 0 1 3 0 1	26 21 18

Problem I. Subset Frequencies

Source file name:	frequencies.c, frequencies.cpp, frequencies.java, frequencies.py
Input:	`standard`
Output:	`standard`

Se tienen dos conjuntos A y S de números enteros positivos, el conjunto A tiene cardinalidad n ($|A| = n$) y el conjunto S tiene cardinalidad k ($|S| = k$), recordar que la cardinalidad de un conjunto es el total de elementos distintos que este contiene.

La tarea a realizar consiste en lo siguiente, para cada uno de los s_i elementos del conjunto S se quiere averiguar cuantos subconjuntos de tamaño dos en el conjunto A tienen una suma de sus elementos igual a el. Para hacer esto aun más interesante, la salida debe cumplir el ordenamiento que se pide en la sección "**Output**"!

Input

La entrada del problema consiste de un único caso de prueba. La primera línea del caso de prueba contiene dos números enteros positivos n y k ($3 \leq n \leq 10^6$, $1 \leq k \leq 10^2$), que representan respectivamente las cardinalidades de los conjuntos A y S. Luego, la segunda línea del caso de prueba contiene los n números enteros positivos a_i del conjunto A ($1 \leq a_i \leq 10^8$, $1 \leq i \leq n$), obviamente se garantiza que los n elementos del conjunto A son diferentes. La tercera línea del caso de prueba contiene los k números enteros positivos s_j del conjunto S ($1 \leq s_j \leq 2 \times 10^8$, $1 \leq j \leq k$), de nuevo se garantiza que los k elementos del conjunto S son diferentes.

Output

Su programa debe imprimir k líneas, cada una de ellas conteniendo dos números enteros positivos s_i f_i ($1 \leq i \leq k$) los cuales representan el valor de la suma de los dos elementos del subconjunto de tamaño dos y la frecuencia de ocurrencia de dichos subconjuntos sobre el conjunto A.

Nota: Las k líneas de la salida del programa deben estar ordenadas de forma descendente con respecto a el valor f_i, en el caso de líneas que tenga el mismo valor de f_i, entonces se debe ordenar de forma ascendente con respecto al valor del s_i.

Example

Input	Output
6 3 6 5 1 4 2 3 7 8 6	7 3 6 2 8 2

Explanation

Para el caso de prueba del ejemplo se tienen los conjuntos $A = \{6,\ 5,\ 1,\ 4,\ 2,\ 3\}$ y $S = \{7,\ 8,\ 6\}$. Para los cuales se tienen tres subconjuntos de tamaño dos cuya suma es igual a $s_1 = 7$, estos son $\{1,\ 6\}$, $\{2,\ 5\}$ y $\{3,\ 4\}$. Se tienen dos subconjuntos de tamaño dos cuya suma es igual a $s_2 = 8$, estos son $\{2,\ 6\}$ y $\{3,\ 5\}$. Por último, se tienen dos subconjuntos de tamaño dos cuya suma es igual a $s_3 = 6$, estos son $\{1,\ 5\}$ y $\{2,\ 4\}$.

Use fast I/O methods

Problem J. Juggling Troupe

Source file name:	troupe.c, troupe.cpp, troupe.java, troupe.py
Input:	`Standard`
Output:	`Standard`

At the national centre for computing and advanced circus skills, technical demonstrations by students are strongly encouraged.

A troupe of n novice performers are at this very moment arrayed in a row attempting to put on a juggling show. Unfortunately, none of them are very confident in their craft, and they are struggling. Thus, as soon as an opportunity presents itself, they will try to reduce their part in the performance to make the task easier.

Whenever a juggler has more than one ball in their possession, they will throw one ball to each of their neighbours. In the case that a juggler does not have a neighbour in some direction, they will simply throw the ball offstage instead. Everybody throws their juggling balls simultaneously. The show ends when no juggler has more than one ball.

See Figure J.1 below for an illustration of this process.

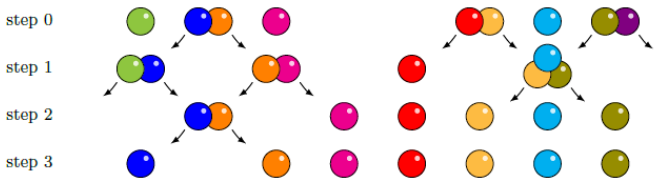

Figure J.1: Illustration of Sample Input 1. A performance with $n = 8$ jugglers.

As a member of the audience, you are not impressed by this performance. However, you do wonder how many balls each of the jugglers will have left at the end of the show.

Input

The input consists of:

- One line with a string s of length n ($1 \leq n \leq 10^6$) over the characters 0, 1 and 2. The ith character in s represents the number of juggling balls initially held by the ith person.

Output

Output a string s of length n over the characters 0 and 1, the ith giving the number of juggling balls the ith person has at the end of the show.

Example

Input	Output
12100212	10111111
000111222000222111222001	111111101111111111111111

Problem K. Knockout Tournament

Source file name:	knockout.c, knockout.cpp, knockout.java, knockout.py
Input:	`Standard`
Output:	`Standard`

Laura is organising a knockout tournament, in which her friend Dale takes part. Laura would like to maximise the probability of Dale winning the tournament by arranging the games in a favourable way. She does not know how to do it, so she asked you for help. Naturally, you refuse to cooperate with such a deplorable act—but then you realise that it is a very nice puzzle!

When the number of players is a power of two, the tournament setup can be described recursively as follows: the players are divided into two equal groups that each play their own knockout tournament, after which the winners of both tournaments play each other. Once a player loses, they are out of the tournament.

When the number of players is not a power of two, some of the last players in the starting line-up advance from the first round automatically so that in the second round the number of players left is a power of two, as shown in Figure K.1.

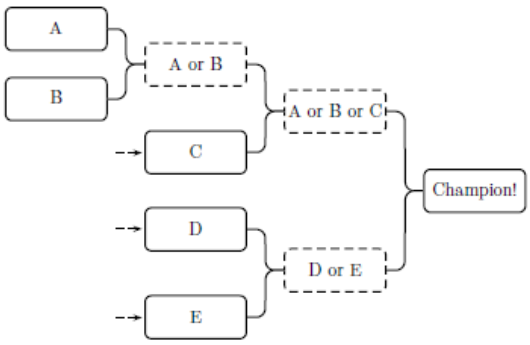

Figure K.1: A tournament tree with 5 players. Players C, D, and E advance from the first round automatically.

Every player has a rating indicating their strength. A player with rating a wins a game against a player with rating b with probability $\frac{a}{a+b}$ (independently of any previous matches played).

Laura as the organiser can order the starting line-up of players in any way she likes. What is the maximum probability of Dale winning the tournament?

Input

The input consists of:

- One line with an integer n ($2 \leq n \leq 4096$), the total number of players.
- n lines, each with an integer r ($1 \leq r \leq 10^5$), the rating of a player. The first rating given is Dale's rating.

Output

Output the maximum probability with which Dale can win the tournament given a favourable setup. Your answer should have an absolute or relative error of at most 10^{-6}.

Example

Input	Output
4 3 1 2 4	0.364285714
5 1 1 3 3 3	0.125