Problem A. Animal Appendages

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

The Town Musicians of Bremen finally arrived in Bremen after their long journey. There, they played music in the streets and had a very successful first day. As night falls, they decide to stop and look for an inn, since taking over a robber's home would be far more difficult in a town.

The Town Musicians of Bremen, made by Gerhard Marcks.
CC BY-SA 4.0 by Wuzur on Wikimedia Commons

Before they can choose an inn to stay at, they need to count the money they earned. They quickly realize that they earned more money than they can count with their paws, hooves, and feet. Kaja, a fellow street musician, notices their dire situation and offers them help. With their human hand Kaja shows the Town Musicians of Bremen how to count in binary from 0 up to 1023 with ten fingers.

The cat, the smartest of their group, notices that this principle can be extended even further. Instead of only extending or curling in fingers or toes, it may also be possible to curl only one of the joints or stretch them into a different direction. Having more than two distinguishable states for some fingers or toes allows them to count even higher.

The number of states fingers or toes can take varies greatly from animal to animal, and even from toe to toe. The cat and the dog both have five toes per paw, but their thumb is barely operable. The rooster has four fingers per foot, but none of them is very flexible. The donkey on the other hand has to get by with hooves only.

Given how many "active" distinguishable states their fingers or toes have, determine the maximum number they can reach counting up from zero. Here "active" states refers to states beyond the neutral state, that is the number of distinguishable gestures excluding a neutral, "do nothing" gesture.

Input

The input consists of:

- One line with ten integers $a_1, \ldots, a_{10}$ ($0 \leq a_i \leq 5$ for each i), the number of different "active" gesture states each finger or toe can represent.

If they have less than ten fingers or toes, there will still be ten numbers in the input, as the remaining numbers will be filled with zero.

Output

Output the maximum number they can count up to from zero with their fingers or toes.

Example

Input	Output
1 1 1 1 1 1 1 1 1 1	1023
0 1 2 3 0 0 4 2 1 0	719
0 0 0 0 1 1 0 0 0 0	3

Problem B. Social Security Benefits

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

The Social Security Administration gives several options to individuals on when to start collecting social security benefits. The sooner you start collecting, the lower the monthly benefits (monthly dollar amount). Since people don't know how long they will live, it is hard to decide when to start collecting, i.e., which option will end up collecting the most?

For this problem, we will assume that there are only two options and that the payment is annual (not monthly). Given the starting age for each option, the annual payment for each option, and an individual's life expectancy (how long the individual will live), determine which of the two options will collect more money.

Input

There are three input lines:

- The first input line provides the starting age for Option$_1$, age_1 ($60 \leq age_1 < 100$), and the annual payment for Option$_1$, $amount_1$ ($200 \leq amount_1 < 10000$).
- The second input line provides the starting age for Option$_2$, age_2 ($age_1 < age_2 \leq 100$), and the annual payment for Option$_2$, $amount_2$ ($amount_1 < amount_2 \leq 10000$).
- The third input line provides the life expectancy, age_3 ($age_1 \leq age_3 \leq 150$).

Output

Print 1 or 2 indicating which option will collect more money. If ties, print 1 (the option that starts collecting earlier).

Example

Input	Output
60 200 65 300 67	1
70 400 85 10000 150	2
60 200 65 10000 63	1
70 300 74 600 78	1
80 200 85 300 80	1

Explanation

In the first Example, Option$_1$ will collect `$1,400` and Option$_2$ will collect `$600`, so Option$_1$ collects more money.

In the fourth Example, both options collect `$2,400` (i.e., tie) so we print 1.

In the last Example, both options collect `$0` (i.e., tie) so we print 1.

Problem C. Single-Elimination Tournament

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

The single-elimination tournaments usually start with a total number of teams that is a perfect power of 2, e.g., the tournament may start with 64 (2^6) teams. Each round, the teams are paired up and half of the teams are eliminated and the remaining half is still a power of 2. The process continues until there is only one team left (the champion).

If the initial team count is not a perfect power of 2, the minimum number of teams play in a "preliminary" round so that the preliminary-round winners plus the remaining teams (teams that got a "bye" round) is a perfect power of 2. For example, if the tournament starts with 20 teams, 8 teams will have to play in the preliminary round so that 4 teams are eliminated and then there will be $12 + 4 = 16$ (a perfect power of 2) teams left. If the tournament starts with 37 teams, 10 teams will have to play in the preliminary round so that 5 teams are eliminated and there will be $27 + 54 = 32$ (a perfect power of 2) teams left.

Given the total number of teams, determine the minimum number of teams that need to play in the preliminary round.

Input

There is only one input line; it provides the total number of teams (an integer between 2 and 10^6, inclusive).

Output

Print the minimum number of teams that need to play in the preliminary round.

Example

Input	Output
20	8
37	10
64	0
1000000	951424
14	12
6	4
256	0

Problem D. Cover Your Bases

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

It's cool to show a friend puzzling equalities such as "$12 + 35 = 28$"! When your friend says "that is not correct", you can say it is correct if 12 is in base 3, 35 is in base 6, and 28 is in base 10!

Given three numbers (X, Y, and S) where S is in base 10, determine the base for X and the base for Y such that $X + Y = S$. Assume that the two bases will be between 2 and 10, inclusive. (Note that a number in base b contains digits 0 through $b - 1$, e.g., a number in base 7 contains digits 0-6.)

Input

There is only one input line; it provides three positive integers: X (1-5 digits), Y (1-5 digits), and S (1-6 digits). Again, S is expressed in base 10.

Output

Print the base for X and the base for Y to make the equality $X + Y = S$ true. Assume that there will be an answer.

If there are multiple correct answers, print the smallest valid base for X. If there are still multiple correct answers, print the smallest valid base for Y.

Example

Input	Output
12 35 28	3 6
5 10 15	6 10
10 5 15	10 6
6 8 14	7 9
10 10 10	2 8
10 10 15	5 10
80 90 170	10 10
10 10 4	2 2

Explanation

For the second Example, any base between 6 and 10 is valid for X but you need to print the smallest valid value.

For the fourth Example, any base between 7 and 10 for X and any base between 9 and 10 for Y will make the equality true but you need to print the smallest valid values.

Problem E. Country Roads

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

The cities of West Bytelandia are well-connected through many roads, but before all cities in West Bytelandia could be connected with roads, the dot.com crash occurred and all construction immediately stopped. Times are good again and there is money left for road construction. The country has decided that it's time once and for all to connect all cities in West Bytelandia with roads so that any city in West Bytelandia is reachable from any other city in West Bytelandia by traveling a sequence of roads connecting cities.

Naturally, the West Bytelandians want to spend as little as possible while achieving their connectivity objective. Help them by writing a program to figure out the least they can spend to develop roads to complete their network so that any city is reachable from any other city by traveling a sequence of roads connecting cities.

Given the number of cities in West Bytelandia, a list of the current roads, as well as a list of potential roads that could be built between pairs of cities along with the cost of each of these potential new roads, determine the least cost necessary to develop some subset of the potential roads given so that any city in West Bytelandia will be reachable from any other city in West Bytelandia via a sequence of roads connecting cities.

Input

The first input line contains three integers: n $\left(2 \leq n \leq 10^5\right)$, indicating the number of cities in West Bytelandia, m $\left(0 \leq m \leq 3 \cdot 10^5\right)$, indicating the number of roads already built in West Bytelandia, and r $\left(1 \leq r \leq 10^5\right)$, indicating the number of new potential roads that could be added in West Bytelandia. The cities in West Bytelandia are numbered from 1 to n.

Each of the next m input lines will contain information about one of the current roads. Each of these input lines will contain two integers, u $(1 \leq u \leq n)$ and v $(1 \leq v \leq n,\ v \neq u)$, indicating that there is a bidirectional road connecting cities u and v, already in place before the dot.com crash. It is possible that more than one road connects the same pair of cities. It is guaranteed that with these m roads, West Bytelandia is not fully connected, i.e., there will exist a pair of cities x and y, such that y is not reachable from x using the m current roads.

Each of the next r input lines will contain information about a new potential road to be built. Each of these input lines will contain three integers, u $(1 \leq u \leq n)$, v $(1 \leq v \leq n,\ v \neq u)$, and c $\left(1 \leq c \leq 10^4\right)$, indicating that a road connecting cities u and v could be built for a cost of c dollars. It is guaranteed that if all r of these roads were added to West Bytelandia, that all pairs of cities in West Bytelandia would be connected by at least one sequence of roads. Note that any two cities can be selected for a new road.

Output

Print the minimum cost of building some subset of the potential roads so that all cities in Bytelandia are connected.

Example

Input	Output
5 5 7 5 2 1 3 3 4 4 1 4 3 1 2 103 1 5 104 3 2 102 3 5 101 4 2 99 4 5 100 3 4 1	99
6 3 2 1 2 3 4 5 6 1 6 6 5 4 9	15

Problem F. Forgotten Fragments

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

Alice wakes up from her dream, convinced that there is more to Wonderland than it seems. The memory is already faint, so she decides to go back to sleep before the dream fades completely.

Alice identified n important dream fragments numbered 1 to n. When she goes back to sleep she will be able to move between fragments via $n-1$ connections that hold all fragments together. However, the evil Queen of Hearts will send her deck of guards to the fragment where Alice starts her dream and make it impassible for the rest of the dream. Alice has just enough time to revisit the events at the starting fragment and escape to a neighbouring fragment before the guards arrive. Then, Alice can move freely between fragments and revisit them as long as she does not go to the starting fragment again.

After Alice wakes up again, she can reason about fragments she revisited but forgets the rest. Each fragment contains a clue and there is a consistent train of thought to unravel the true meaning of Wonderland. She will follow the clues until they lead to a fragment she did not revisit. For example, if she revisited fragments numbered 1, 2, 3, 5, and 6, she can follow the truth until the clue of the 3rd fragment.

How much of the true meaning of Wonderland can Alice unravel? Since Alice does not know where her dream will start, find the answer for all possible starts.

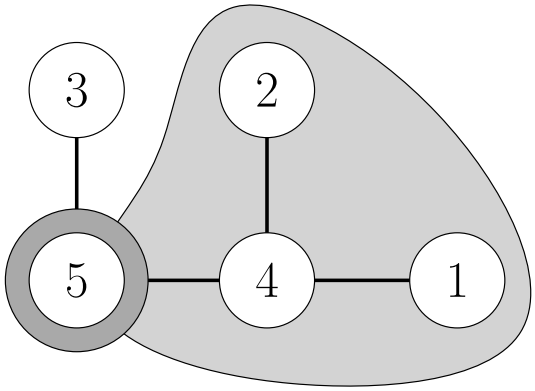

Figure 1. Visualization of Example Input 1. The dream starts in the fragment 5. Alice escapes to the subtree marked in gray to revisit clues 1, 2, 4, and 5. She cannot reach clue number 3 and only deciphers the meaning of the first two clues.

Input

The input consists of:

- One line with an integer n ($2 \leq n \leq 2 \cdot 10^5$), the number of dream fragments.
- $n-1$ lines with two integers u, v ($1 \leq u, v \leq n, u \neq v$), representing a connection between the dream fragments with the uth clue and vth clue.

It is guaranteed that each dream fragment is connected directly or indirectly to every other dream fragment.

Output

Output a line with n integers $a_1, \ldots, a_n$ ($1 \leq a_i \leq n$).

The value a_i should be the number of clues Alice can decipher if she starts her dream at the fragment with number i.

Example

Input	Output
5 3 5 5 4 2 4 4 1	5 5 5 1 2
5 1 4 1 5 2 5 3 5	3 5 5 5 1

Problem G. Grimms' Fairy Tales

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

Bothers Grimm are excited to show you their newest project. They already finished their famous fairy tale bundles with over 200 fairy tales in total. This year they wanted a new challenge: programming problems incorporating various fairy tales. Over months and months they searched far and wide to gather the best fairy tale programming problems. The brothers want to publish their new bundle of 13 programming problems before the month of February arrives. With a few days to spare, the whole bundle was finished. Except, you, the editor noticed an important detail: they forgot to add page numbers in the table of contents. Please help them by finding out the right page number for each problem title.

```
Problems

A Animal Appendages
B Social Security Benefits
C Single-Elimination Tournament
D Cover Your Bases
E Country Roads
F Forgotten Fragments
G Grimms' Fairy Tales
H Who wants to be a Millionaire?
I World Population
J Jaded Journey
K Knavish Knockout
L What's the Order Anyway?
M Picture Caption
```

The table of contents without page numbers.

Input

The input consists of:

- One line with a string s ($1 \leq |s| \leq 100$), the name of one of the problems in this contest.

The string s consists of English lowercase and uppercase letters (`a-z` and `A-Z`), apostrophes (`'`) and spaces. It is guaranteed that the title is the exact name of one of the problems in this contest, and is written exactly as in this document, including spacing and capitalization.

Output

Output the page number where the problem statement starts.

Example

Input	Output
Animal Appendages	1
Grimms' Fairy Tales	10
Picture Caption	21

Problem H. Who wants to be a Millionaire?

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

In the popular game show, "Who wants to be a Millionaire?," contestants are given 15 questions, in increasing order of difficulty. The cash a contestant wins is dependent on the number of questions they answer correctly. An incorrect response immediately ends the game, and the contestant either wins no money, or a reduced amount of money. Alternatively, a contestant could choose to cash out at any time and get all the winnings for their correctly answered questions.

In this problem, we'll analyze an easier version of the game where an incorrect response to a question automatically means no winnings.

In this version of the game, a contestant will receive up to n questions in series. Let the questions be numbered 1 through n. For each question, there is a probability that the contestant will answer the question correctly. After listening to each question, the contestant can choose to answer the question or to cash out. If the previous question was answered correctly (and it wasn't the last question), the contestant gets to hear the next question, at which point they have the same choice to make, i.e., choose to answer the question or cash out.

To help the contestant, they get to "ask an expert" for help on two questions, i.e., the contestant can get help from an expert on two questions out of the n questions. For each question, there is an alternate probability the expert will correctly answer the question (and this is guaranteed to be strictly higher than the probability the contestant will answer the question correctly). On any two questions out of the n total questions, the contestant can choose this option once they've decided to answer the question.

For each of the n questions, there is a dollar amount assigned. That is the amount the contestant wins if that particular question was the last question they answered correctly. For example, if after hearing question `#8`, a contestant decides to not answer it and cash out, the amount of money they will win is the amount of money associated with question `#7`, the last question they answered correctly. Recall that if they answer their last question incorrectly, the contestant wins no money.

Given the number of questions in the game, the dollar amount assigned to each of the questions, the probability that the contestant will correctly answer each of the questions, and the probability that the expert will correctly answer each of the questions, determine the maximum expected winnings of the contestant, if they play optimally.

Input

The first input line contains a single integer, n ($1 \leq n \leq 100$), indicating the number of questions in the quiz show.

The second input line contains n integers, w_1, w_2, ... w_n. w_i equals the total winnings for answering question i correctly and then stopping the game. It is guaranteed that $0 < w_1 < w_2 < w_3 < \ldots < w_{n-1} < w_n \leq 10^9$.

The third input line contains n floating point numbers, p_1, p_2, ... p_n. p_i equals the probability of the contestant answering question i correctly. It is guaranteed that each of these will be specified to exactly 2 decimal places and $0 \leq p_i < 1$.

The fourth input line contains n floating point numbers, q_1, q_2, ... q_n. q_i equals the probability of the expert answering question i correctly. It is guaranteed that each of these will be specified to exactly 2 decimal places and $p_i < q_i \leq 1$.

Output

Print the maximum expected winnings of the contestant, if they play optimally. Any answer within an absolute or relative error of 10^{-6} of the correct answer will be accepted.

Example

Input	Output
2 10 100 0.25 0.01 0.99 1.00	99.0
3 100 101 102 0.90 0.80 0.70 1.00 0.81 0.71	100.0

Explanation

For the first Example, the best strategy is to ask the expert both questions.

For the second Example, the best strategy is to ask the expert the first question and then stop.

Problem I. World Population

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

According to available data, the world population around the start of each century was roughly:

- 1st `Century`: around 300 million people
- 12th `Century`: around 450 million
- 17th `Century`: around 500 million
- 18th `Century`: around 800 million
- 19th `Century`: around 1.2 billion
- 20th `Century`: reaching 2 billion by the mid-century, then rapidly increasing to over 6 billion by the end
- 21st `Century`: currently around 8 billion people, projected to reach around 10 billion by mid-century

Some researchers are studying the population growth in various countries in the world.

We assume there are n countries (numbered 1 through n) that are in a straight line, i.e., Country_2 is next to Country_1, Country_3 is next to Country_2, and so on. Given the initial population for each country and population growth for different countries, you are to determine the total population for different range (contiguous set) of countries.

Input

The first input line contains an integer, n $\left(1 \leq n \leq 5 \cdot 10^5\right)$, indicating the number of countries. The following n input lines provide the initial population for these n countries. Each line will be an integer between 1 and $15 \cdot 10^8$, inclusive; first integer is the population for the first country, second integer is the population for the second country, etc.

After the initial population information for all the countries, the input will have a set of transactions (operations) to be processed. This section of the input starts with an integer, t $\left(1 \leq t \leq 10^5\right)$, indicating the number of transactions. Each of the next t input lines contains a transaction to be processed. There will be two types of transactions:

- Update: A given country's population grows/decreases by an amount, e.g., Country_{17} grows by 500. This input line starts with the letter `U` in the first column, followed by one space, followed by a valid country number, followed by a space, followed by an integer less than or equal to 10^9, indicating the increase/decrease in the given country's population. Assume that a country's population will never go below zero (if the integer on this input line is negative).
- Retrieve: This input line starts with the letter `R` in the first column, followed by one space, followed by a valid starting country number, followed by a space, followed by a valid ending country number. The total population for this range (contiguous set of countries) is being requested, e.g., countries 10-18. Assume that the ending country number will not be less than the starting country number, i.e., the requested range is valid.

Output

There is no output required for the Update transactions. For each Retrieve transaction, output a separate line providing the total population for the requested range.

Example

Input	Output
5 10 20 15 30 25 8 R 2 4 U 2 25 U 3 40 R 2 4 R 1 5 U 2 15 U 4 -20 R 1 5	65 130 165 160

Problem J. Jaded Journey

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

Painting of a baghlah, a large deep-sea traditional Arabic sailing vessel. CC BY-SA 4.0 by Xavier Romero-Frias on Wikimedia Commons

Hindbad's life was transformed during the seven days he spent with Sindbad the Sailor. As he listened to the stories of Sindbad's seven voyages of distant seas and strange lands, an unfamiliar world unfolded before his eyes. Wanting to repay Sindbad for his generosity and companionship, Hindbad resolved to take him on one final journey, Sindbad's eighth voyage.

This final journey entails an n farsakh[1] long sea travel. As Hindbad was not nearly as financially endowed as his companion, the ship he hired for the voyage had some clear deficiencies. To make matters worse, Sindbad's reputation as the lone survivor of so many voyages had preceded him. The crew, convinced they were ferrying a walking ill omen, demanded unreasonable extra compensation in exchange for letting him sail with them.

The ship has two means of moving. The crew can row the ship with rudders, but for each farsakh they row they demand x additional food ration packs for their effort. When there is wind, the sail can be used to move instead. Using the sail to travel will not cost Hindbad anything, but since he hired a cheap ship, the sail breaks down frequently. He can ask the crew to fix the sail for r extra ration packs as often as necessary. Unfortunately, whenever the sail is repaired, it will break after d further farsakhs, regardless of how much it is used during this time. Initially, the sail is broken.

Given an accurate forecast on where there will be wind and where there will not, determine the minimum number of extra food ration packs Hindbad needs to load onto the ship.

Input

The input consists of:

- One line with four integers n, x, r, and d ($1 \le d \le n \le 2 \cdot 10^5, 1 \le x, r \le 10^9$), the distance you need to travel, the required extra food packs for rowing one farsakh, the required food packs to repair the sail, and the distance until the sail breaks after repair.
- One line with n integers $w_1, \ldots, w_n$ ($0 \le w_i \le 1$ for each i), with w_i being 1 if there is wind to use the sail in the ith farsakh of the journey, and 0 otherwise.

Output

Output the minimum extra food ration packs Hindbad needs to load onto the ship.

Example

Input	Output
3 2 1 3 0 0 0	6
4 3 5 2 1 0 1 1	11
7 2 3 3 0 1 1 1 1 0 1	10

[1] An old distance unit used in the Middle East, roughly equivalent to 6 km

Explanation

For the example 1, since there is no wind in this sample, the crew has to row all three farsakhs. Therefore the total extra food ration packs needed is $3 \cdot x = 6$.

For the example 2, here it is optimal to let the crew row the first two farsakhs and then repair the sail for the last two farsakhs.

Problem K. Knavish Knockout

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

Richilde selling her apples.
Public Domain on Wikimedia Commons

Richilde has just received shocking news from her magic mirror. Not only is Blanca, her unbeloved stepchild, alive and well, but on top of that, she is still the fairest of them all! Richilde, who has already made several failed attempts to get rid of Blanca using poisoned soaps and letters, is furious. This time, she decides to take care of the problem on her own.

Richilde knows that Blanca lives in a cottage in a far away forest together with $n-1$ dwarfs. She plans to travel to this forest to execute the following plan: for k consecutive days, Richilde will visit the cottage disguised as an old peddler and sell n poisoned apples for them to eat at their dinner.

With help from her court physician Sambul, Richilde has collected $n \cdot k$ apples from his special apple tree, which bears fruit ingrained with a slow acting poison. Each apple has a certain size and contains a certain amount of poison. Since dwarfs are known for their hearty appetite and since she knows Blanca to be quite modest, Richilde is confident that each day at dinner, Blanca will eat the smallest apple that was sold on that day. The remaining $n-1$ apples will be eaten by the dwarfs. Of course, Richilde wants to maximize the total amount of poison that is consumed by Blanca. To achieve this, she will carefully select which n apples to sell on each of the k days.

Unbeknownst to Richilde, Sambul has long ago ensured that all poisonous soaps, letters and apples that Richilde might use in her wicked plans are actually laced with a non-lethal sleeping poison. He is therefore confident that Blanca will yet again survive Richilde's latest attempt. However, he is still worried about Blanca's health and wonders how much of the sleeping poison she will consume in total. What is the maximum total amount of poison that Blanca will consume, given that Richilde can choose which apples to sell on each day?

Input

The input consists of:

- One line with two integers n, k ($1 \leq n, k \leq 10^5$, $n \cdot k \leq 2 \cdot 10^5$), the number of apples sold each day and the number of days.
- $n \cdot k$ lines, the ith of which contains two integers s_i, p_i ($1 \leq s_i, p_i \leq 10^9$), the size and poison amount of the ith apple.

It is guaranteed that all sizes s_i are distinct.

Output

Output the maximum total amount of poison that will be consumed by Blanca (the total amount is the sum over the poison values of all apples she will eat).

Example

Input	Output
2 3 1 10 2 10 3 10 6 5 7 5 8 5	30
3 3 1 8 2 5 9 4 8 8 6 7 7 4 3 8 4 4 5 6	23

Problem L. What's the Order Anyway?

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

Alice, Bob, Carol, Denise, Eddie and Frank are up to their usual tricks. They're performing in the local comedy show but haven't told you the order in which they are presenting their stand-up routines. Instead, they've given you cryptic clues such as (with A standing for Alice, B for Bob, C for Carol, D for Denise, E for Eddie and F for Frank):

- $A > B$
- $F < A$
- not C D

The first clue means that A is performing before B. The second clue means that F is performing after A. The third clue means that C and D are not performing consecutively. (In essence, if we were treating the acts as numbers, the inequality signs are assuming that the acts are sorted from greatest to least.)

Given the above clues, a possible order for the acts is Alice, Carol, Bob, Denise, Frank, Eddie.

What you've realized is that even with their clues, it might be impossible to pin down the exact order of their acts. Thus, you've settled for figuring out how many orderings are possible given the clues they've given you.

Given clues about the order of comedy acts, determine the number of valid possible orderings for the acts.

Input

The first input line contains two space-separated integers: n ($2 \leq n \leq 10$), indicating the number of comedy acts, and c ($1 \leq c \leq 10$), indicating the number of clues you've been provided. The acts are denoted by the first n uppercase letters.

The clues are provided in the following input lines, one clue per line. Each of these input lines has three space-separated pieces of information: a ($1 \leq a \leq 3$), x and y, as described below:

- x and y are guaranteed to be distinct uppercase letters out of the first n letters,
- a represents the type of restriction where 1 indicates that x's act comes before y's, 2 indicates that x's act comes after y's and 3 indicates that x's act and y's act don't occur consecutively (in either order).

It's guaranteed that the input clues won't be contradictory, and that there will be at least one valid ordering of the comedy acts.

Output

Print the number of different orders in which the comedy acts could be.

Example

Input	Output
6 3 1 A B 2 F A 3 C D	160
3 2 1 C A 2 B A	1

Problem M. Picture Caption

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

The UCF Programming Team Lab (PTL) is located in HEC. The team photos are posted outside this lab. When preparing the caption for a picture, there is always the question of how many names should be listed on each line under the picture.

Given the number of names, number of lines to use for the picture caption, and number of letters in each name, we would like to minimize the length of the longest line in the caption.

Note that the names must appear in the caption in the order of the input, i.e., we don't want to reorder the names. Also note the following common constraints:

- No space before the first name on each caption line.
- No space after the last name on each caption line.
- Exactly one space between two consecutive names on each caption line.

Input

The first input line contains two integers n and k $\left(1 \leq k \leq n \leq 10^5\right)$, indicating (respectively) the number of names and the number of lines to use for the picture caption.

The second input line contains n integers c_i $\left(1 \leq c_i \leq 10^4\right)$, representing the number of letters in each name in order.

Output

Print the length of the longest line in the caption, keeping in mind that we would like to minimize this value.

Example

Input	Output
7 4 1 2 8 3 5 2 7	10
7 5 1 2 8 3 5 2 7	8
7 2 3 1 1 3 9 5 2	18