CMU High School Programming Contest

March 7, 2020

The Problems:

1. Seven Hundred Thousand Segments
2. Vigenere Cipher
3. Majority Queries (Small)
4. Majority Queries (Large)
5. Chicken Nuggets
6. Crossing a River
7. The LOL Game (Small)
8. The LOL Game (Large)

Notes:

The contest judge is located at `http://contest.cs.cmu.edu:8080`

All problems are weighted equally. The goal is to solve as many problems as possible.

The problems are of varying difficulty, and they have been randomly ordered. Viewing the scoreboard can help you classify problems in terms of difficulty.

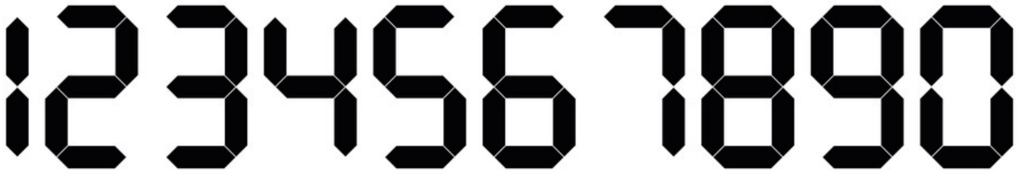
The duration of the contest is three hours. During the last thirty minutes of the contest the scoreboard is "frozen", that is, it stops being updated as a result of new submissions.

Good Luck!

Problem 1. Seven Hundred Thousand Segments

Time limit:	1 second
Memory limit:	256 megabytes

A 7-segment display element can present any of the ten decimal digits by illuminating a subset of the seven segments, as shown in the following diagram.



So, for example, the digit 1 has two segments illuminated, and the digit 8 has all seven illuminated.

There are n 7-segment display elements in a row, displaying a given n-digit number. Among all n-digit numbers that can be displayed with the same number of illuminated segments, which one is the largest? Write a program to compute this.

Input

The first line contains a single integer T, the number of test cases. This is followed by T lines, each containing a single integer n followed by a space then followed by n decimal digits representing the number currently displayed. The sum total of all the ns in the input is at most 10^5.

Output

For each test case print the answer on a single line.

Example

standard input	standard output

4
1 3
3 512
3 079
35 11111111111111111111111111111111114
	5
	977
	997
	77111111111111111111111111111111111

Problem 2. Vigenere Cipher

Time limit:	1 second
Memory limit:	256 megabytes

A Caesar cipher is a method of encrypting text. Given a message, the cipher generates a 'ciphertext' by shifting all the letters in the message by a given number. For example, if the string "DOG" were to be shifted by 2, then 'D' would become 'F', 'O' would become 'Q', and 'G' would become 'I' resulting in the string "FQI". If a shift goes past the end of the alphabet, it wraps around back to 'A'. For example, 'Z' shifted by 3 would become 'C'. Then, if one knows the shift amount, one can decrypt a ciphertext by simply shifting each letter the reverse direction.

A more advanced type of cipher is a Vigenere Cipher. In a Vigenere Cipher, in addition to the message, there is also a "key", which is another string of letters. Each letter in the key corresponds to a different shift amount, with 'A' corresponding to 0, 'B' corresponding to 1, and so on up to 'Z' which corresponds to 25. To encrypt a message using the key, we first concatenate the key onto itself until its length is equal to (or greater than) the length of the message. For example, if our message was "HELLOWORLD" and our key was "CMU", we would repeatedly add "CMU" onto itself until we got "CMUCMUCMUC". Then, we encrypt each letter in the message using a Caesar cipher with shift amount associated with the corresponding letter in the key. Using our previous example, the first letter of the message, 'H', will get encrypted with a shift of 2 since the first letter in the key, 'C', corresponds to a shift of 2. Likewise, the next letter 'E', will get encrypted with a shift of 12 since the next letter in the key, 'M', corresponds to a shift of 12. This continues for the entire message to get the final string "JQFNAQQDFF". To decrypt a vigenere cipher given the key, one only needs to decrypt each individual letter as they would decrypt a Caesar cipher, since the key tells them the shift amounts for each letter.

Your task is to write a program that can encrypt and decrypt a Vigenere Cipher.

Input

The input consists of 3 lines. The first line contains a single character 'E' or 'D' corresponding to "Encrypt" and "Decrypt" respectively. The next line contains a non-empty string of length at most 1000 consisting only of the characters A-Z, representing the message or ciphertext. The last line contains another a non-empty string of length at most 1000 consisting only of the characters A-Z, representing the key.

Output

Output a single line, the resulting string from the encryption or decryption.

Examples

standard input	standard output

`E`
`HELLOWORLD`
`CMU`
	`JQFNAQQDFF`

`D`
`JQFNAQQDFF`
`CMU`
	`HELLOWORLD`

`E`
`ABCDE`
`A`
	`ABCDE`

Problem 3. Majority Queries (Small)

Time limit:	1 second
Memory limit:	256 megabytes

You're given a string S of length n consisting of `A`s and `B`s. You are then asked to answer several queries about the contents of the string.

The queries are of the form (i, j) with $0 \leq i \leq j < n$. Each query represents a substring of S from index i to index j (inclusive). Your task is to determine which character occurs more within the substring given by each query.

Note that zero-based indexing is used in S. So the first character of S is denoted $S[0]$ and the last one is $S[n-1]$.

Input

The first line contains n, q, $(1 \leq n, q \leq 100)$, the length of the string, and the number of queries. The next line contains the string S. Then the next q lines each contain two integers, i, j, $(0 \leq i \leq j < n)$, the query points.

Output

Output n lines. On each line, for the corresponding query, if there were more `A`s than `B`s, output "`A Wins`". If there were an equal number `A`s and `B`s, output "`Tie`". If there were more `B`s than `A`s then output "`B Wins`".

Examples

standard input	standard output
5 4 ABBAA 1 2 3 4 1 4 1 3	B Wins A Wins Tie B Wins
10 2 BABAABBBAB 2 7 2 3	B Wins Tie

Problem 4. Majority Queries (Large)

Time limit:	1 second
Memory limit:	256 megabytes

This is the same as the previous problem except that the string S of `A`s and `B`s can have length up to 100,000, and the number of queries can be up to 100,000. The time limit is still 1 second.

Example

standard input	standard output
1 1 B 0 0	B Wins

Problem 5. Chicken Nuggets

Time limit:	10 seconds
Memory limit:	256 megabytes

Steven wants to order some chicken nuggets from McTartan's, CMU's famous fast food restaurant. Steven is very particular, and wants to get exactly n nuggets. Nuggets at McTartan's come in k different order sizes. However, McTartan's charges a flat fee for every order of nuggets, so Steven also needs to minimize the total number of orders. Help Steven figure out the minimum number of orders he can make so that he gets exactly n chicken nuggets.

Input

The first line of input consists of two integers, n, k, with $1 \leq n \leq 10^4$ and $1 \leq k \leq 1000$. The next line contains k unique integers, each between 1 and 1000, representing the possible order sizes.

Output

Output a single integer, the minimum number of orders needed to get exactly n nuggets. If it is not possible to get n nuggets, output the string "Impossible".

Examples

standard input	standard output

`28 3`
`4 10 20`
	`3`

`25 2`
`2 4`
	`Impossible`

Problem 6. Crossing a River

Time limit:	1 second
Memory limit:	256 megabytes

Mark is travelling on the Oregon trail. Besides getting dysentery, he also needs to cross a river. A river is defined by the region between two lines, in the xy-plane. The south side is the line $y = 0$ and the north side is the line $y = w$.

Mark starts anywhere on the south side of the river, and he wants to get to the north side. There are some stones in the river that he can step on. He can step a distance of s (or less) from one stone to another, or between a stone and a river bank.

Help Mark determine if it's possible to cross from the south side of the river to the north side, and if so compute a sequence of stones he can use to achieve this.

Input

The first line contains three integers n, w, s $(1 \le x_i \le 10^4, 1 \le s < w \le 10^4)$, the number of stones, the width of the river, and the maximum distance Mark can step. Following this are n lines, the ith of which contains (x_i, y_i), $(-10^4 \le x_i \le 10^4, 0 < y_i < w)$ the position of the ith stone. The stones are numbered $1, 2, \ldots, n$, and are all at distinct locations strictly within the river. The coordinates are all integers.

Output

Output a single line line containing either "YES" or "NO" indicating whether Mark is able to cross the river. If the answer is "YES", output a second line containing a sequence of stone numbers allowing Mark to cross the river. These numbers must all be distinct, and be between 1 and n (inclusive). The distance between any consecutive stones (and the distance between the first/last stones and the shore) must be at most s.

Examples

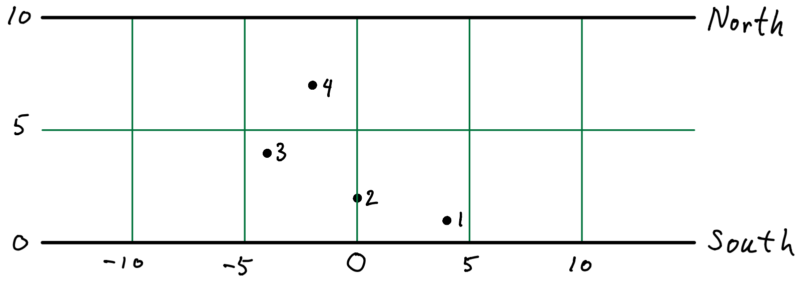
standard input	standard output

4 10 4
4 1
0 2
-4 4
-2 7
	YES
	3 4

4 10 3
4 1
0 2
-4 4
-2 7
	NO

Note

The only difference between these two test cases is the stride length. They are illustrated below.



Problem 7. The LOL Game (Small)

Time limit:	5 seconds
Memory limit:	256 megabytes

To compete with League of Legends, CMU is making their own `LOL` game. The `LOL` game is played as follows. Start with a blank piece of paper with a row of n empty boxes on it. The players alternate filling in a blank box with a letter `L` or a letter `O`. Once a box is filled, it remains the same and can never be used again. The goal is to be the first player to form a pattern `LOL` among three consecutive boxes.

Assuming both players play with an optimal strategy for themselves, the result of the game is defined, and can be a win for the next player to play, a draw (nobody wins), or a win for the previous player. We denote these three possibilities with the letters N for the Next player having a winning strategy, D for a Draw, and P for Previous player having a winning strategy. We say the value of a position is N, D, or P as defined here.

For example, suppose the state of the game is `LL-L`. (The "`-`" symbol indicates an empty box.) Then the next player puts a `O` in the blank and wins the game. Therefore `LL-L` is an N position. On the other hand, consider `L--L`. Now no matter what the next player does, the other player wins, so `L--L` is a P position.

In this problem you are to write a program that can compute the value of any position that might occur in an `LOL` game of limited size.

Input

The first line consists of a single integer n, the number of boxes in the game. The next line is a sequence of n characters from the set $\{$`L`,`O`,`-`$\}$. A "`-`" indicates an empty box. Here n is at most 26 and the number of "`-`" characters is at most 7. It is guaranteed that the position is not a terminal one. In other words it will not contain the winning `LOL` pattern, and it will contain at least one "`-`".

Output

Output a single character `N`, `D`, or `P`, the value of game at the given board position.

Examples

standard input	standard output
4 LL-L	N
4 L--L	P
7 -------	N

Note

In the last example, a winning move for the next player is to put an `L` in the middle box.

Problem 8. The LOL Game (Large)

Time limit: 10 seconds
Memory limit: 256 megabytes

This problem is the same as the previous one except that the number of "-" characters can be up to 13. In the previous problem the limit was 7. Also the time limit is 10 seconds as opposed to 5 seconds.

Examples

standard input	standard output

`26`
`LOOOOOOOOLLLL-------------`
	`N`

`18`
`O---O---O---O---L-`
	`D`

`21`
`---L--L---O-L---LOOLL`
	`P`