XXXI Programming Contest ICPC-UCV 2025

Contest Session

Sponsored by

Caracas, October 11th 2025

Problem A
Deficient, Perfect, Abundant

Source file name: `dpa.cpp`, `dpa.py`

The sum $S(N)$ of a number N, is defined as the sum of all of its proper divisors, that is, all the numbers that divide N except for itself. For example, $S(15) = 9$. The proper divisors of 15 are $\{1, 3, 5\}$, so $S(15) = 1 + 3 + 5 = 9$

We say that a number N is *Deficient* if $S(N) < N$, *Perfect* if $S(N) = N$ and *Abundant* if $S(N) > N$.

A few examples:

- 15 is a deficient number, because $S(15) = 1 + 3 + 5 = 9$.
- 28 is a perfect number, because $S(28) = 1 + 2 + 4 + 7 + 14 = 28$.
- 20 is an abundant number, because $S(20) = 1 + 2 + 4 + 5 + 10 = 22$.

Your task is simple. Given a integer N you must tell what kind of number N is (either deficient, perfect or abundant). Furthermore, you must find the smallest *abundant* number greater than N. For example, given $N = 15$ we find that 15 is a deficient number, and the next abundant number would be 18, because $S(18) = 1 + 2 + 3 + 6 + 9 = 21$

Input

The input consists of one line with a single integer N ($1 \leq$N$\leq 10^{12}$).

The input must be read from standard input.

Output

You must print one line with one character 'D', 'P' or 'A' depending if N is deficient, perfect or abundant respectively, followed by an integer A, which is the smallest abundant number that is greater than N. Both character and number must be separated by exactly one space.

The output must be written to standard output.

Sample input	**Sample output**
`15`	`D 18`

Sample input	**Sample output**
`28`	`P 30`

Sample input	**Sample output**
`453225641386`	`D 453225641388`

Problem B
Midnight Craving

Source file name: `midnight.cpp`, `midnight.py`

Finsbury Park is a beautiful park located in North London, and as many parks in London you can see all sort of things like trees, squirells, birds, happy families, dogs, poop, litter, people trading illegal stuff, etc.

The park is relatively safe during the day, however it is a completely different story at night. Many people avoid it late at night, except for delivery guys! The park represents a massive shortcut for food deliveries, and London is well known for its midnight cravers (gotta love that good old 3 am maccies) so there's a big market out there!

The park is full of lamp posts, but unfortunately some of these lamps are broken, and they switch off and on intermittently. Thieves lurk in the dark areas waiting for victims to rob, while illuminated areas are always safe to traverse.

For the sake of simplicity, we will represent the park as a one-dimensional binary array A of length N, where a 0 represents a spot with a turned-off lamp (a dark spot) and a 1 represents a spot with a turned-on lamp (an illuminated spot)

Tonight you will help Johnny to complete D deliveries! Johnny knows that for each of this deliveries he has to traverse a contiguous subsegment of the park. Specifically, to complete the delivery i, he knows he has to traverse the park segment starting at the spot l_i and ending at r_i. Additionally, Johnny knows that during the night some subsegments of the park will alternate the state of all of its lamp posts. He knows this will happen exactly U times, and for the event i he knows that the lamp posts starting at l_i and ending at r_i will alternate their state (that is, all lamps that were on will be turned off, and all lamps that were off will be turned on).

For each of the D deliveries Johhny has to complete, you must tell him what's the longest contiguous not illuminated area he has to cross, taking into account the state of the lamps by the time he is doing the delivery.

You will receive the deliveries and lamp updates specifications as a list of Q queries.

Input

The input will start with two integers N and Q ($1 \leq N, Q \leq 2 \cdot 10^5$) separated by a single space. N will be the length of the park, and Q will be the number of deliveries and updates that Johnny has to complete.

The next line will contain a binary string of length N which represents the park. A 0 indicates a turned-off lamp and a 1 a turned-on lamp.

The following Q lines will indicate either a delivery to be completed, or a lamp update. They will follow the format:

- 1 L R: A delivery must be made, and Johnny must traverse the park from L to R
- 2 L R: The lamp posts located in the interval $[L, R]$ will alternate their state

For each of these queries, it is guaranteed that $1 \leq L \leq R \leq N$.

The input must be read from standard input.

Output

For each entry on the input that represents a delivery, you must print a line with a single integer, the length of the longest not illuminated segment that Johnny has to traverse. Don't print anything for the updates.

The output must be written to standard output.

Sample input

```
10 8
1111111111
1 1 10
2 3 5
1 1 10
2 4 4
1 1 5
2 1 10
1 1 10
1 1 5
```

Sample output

```
0
3
1
5
2
```

Sample input

```
8 6
01001010
1 1 8
2 5 5
1 1 8
2 4 6
1 4 7
1 3 8
```

Sample output

```
2
4
0
1
```

Problem C
Enemy Communications

Source file name: `enemy.cpp`, `enemy.py`

The intelligence service has intercepted a set of encrypted enemy communications. Your task is to decrypt each message using a decryption key specific to each group of messages. You will be provided with a constant base string containing all the possible characters that make up the encrypted messages:

_*ABC!DEF,GHIJ?KLM.98765'43210NOPQRSTUVWXYZ-$

This string is 45 characters long and is considered cyclical, meaning that the character after the last '$' is the first '_'. Indexing for this string starts at 1.

Decoding process (each communication line) consists of 2 consecutive steps for each group of communications to be decoded:

- Step 1 (Simple Cyclic Shift): Each character in the encoded message (except for spaces) must be replaced by the immediately following character in the base string (using the cyclical nature as many times as necessary), respecting the UPPER/lower case of the original character (except for digits and special characters, where UPPER case is assumed for the resulting character). Thus, the character 'a' would be changed to 'b', and the character '$' would be changed to '_', and '!' to 'D'; for example.
- Step 2 (Shifting by Guide Key): After Step 1 of the pre-encoding process, the resulting string undergoes a second transformation using a key sG phrase (Guide) provided in the input, an effort of multiple souls who have obtained them, hoping for your abilities to properly decode the intercepted communications. If an exact match (UPPERCASE/lowercase) is found (from left to right) within the sG Guide phrase (at position d in this phrase) in the pre-processed phrase from Step 1, said resulting character must be replaced by the corresponding character from the base string with a shift d (cyclically), where d is the position (index, starting at 1) of the first occurrence of that character in the Guide phrase. If this character is not found in the Guide phrase, the resulting character must not be altered. For example, if our Guide phrase is "uz?*xQ!u_u_u"', and after processing the initial phrase "mmmsn VC0_" to "...to W!N*" , in the Guide phrase we will find the characters '*' and '!' at positions d=4 and d=7, respectively. We then replace them with their original positions with these offsets (even if they are cyclical) in the base string, leaving: ...to WIN! The spaces ' ' are not modified in either step.

Input

The first line contains an integer $N(0 < N < 10^3)$ representing the communication group numbers to be processed. Then, several groups (N) with the information to be processed, each group starting on a line with an integer $F(0 < F < 10^2)$ representing the number of phrases the group contains; and then, separated by a space, a string sG : the Guide string (max. 104 characters). Finally, F lines of sentences to be decoded (max. 104 characters per line).

The input must be read from standard input.

Output

Print as many groups (N) as indicated in the input, each starting with its number in brackets from [1] to [N], each group followed by F lines of the decoded sentences; and finally, a separator line between them.

The output must be written to standard output.

Sample input	**Sample output**
`2`	`[1]`
`2 uz?*xQ!u_u_u`	`We NeeD to prActice More...`
`Pd Odd! sn oq*bshbd Lnqdmmm`	`...to WIN!`
`mmmsn VCO_`	
`1 cvbzdScX.MnMbcUnXxb7bvV`	`[2]`
`SGT TO!`	`THE END`

Problem D
Yummy

Source file name: `yummy.cpp`, `yummy.py`

Johan constantly receives questions about how *yummy* a food is. He figured a way to encode all the ingredients of a particular dish in a string in such a way that when the number of “Yu” found in the string are bigger than the number of characters in the string (without the “Yu”) divided by 2, then that food is considered *yummy*. Your task is to help Johan to automate this task for any given string.

Input

A single string of size N ($0 < N \leq 2^{10}$).

The input must be read from standard input.

Output

For each string print “Yummy” if the condition from Johan is met otherwise print “Not so Yummy”.

The output must be written to standard output.

Sample input	**Sample output**
`YukYuk?`	`Yummy`

Sample input	**Sample output**
`Yummy`	`Not so Yummy`

Problem E
Infection Lethality

Source file name: `infection.cpp`, `infection.py`

You have been hired as a cybersecurity advisor by a major corporation. Their concern is simple but alarming: if a single computer in their infrastructure is compromised, how devastating could the consequences be? Your task is to determine the lethality of the infection: given the description of the network and the identity of the initially compromised computer, calculate how many computers in total would ultimately be affected.

Input

The input starts with a line containing an integer N ($1 \leq N \leq 10^5$) indicating the number of computers in the network, followed by another line with an integer M ($1 \leq$M$\leq 2 \times 10^5$), followed by another line with an integer Q ($1 \leq$Q$\leq 10^5$). Then you will find M lines each containing two integers u and v representing a bidirectional connection between computers u and v. Finally you will find Q lines, each one representing a query over the network; each query consists of an integer S, which identifies the initially infected computer for that query.

The input must be read from standard input.

Output

You must print Q integers one per line, representing the total number of computers infected for each query (including the initially compromised one).

The output must be written to standard output.

Sample input	**Sample output**

```
5
4
2
1 2
1 3
1 4
1 5
1
3
```
	```
	5
	5
	```

Sample input	**Sample output**
`8`	`3`
`6`	`4`
`3`	`1`
`1 2`	
`2 3`	
`4 6`	
`4 5`	
`6 7`	
`5 7`	
`1`	
`6`	
`8`	

Problem F
The Rebellious Boxes Challenge

Source file name: `boxes.cpp`, `boxes.py`

At the International Institute of Liquid Containment (IILC), a group of scientists has spent years studying how different volumes of water react when foreign objects are inserted into hermetic boxes. The official goal is "to prevent catastrophes in high-pressure experiments" although many suspect that in reality they only do it to torture new interns.

Each IILC box is a rectangular parallelepiped with base dimensions N (length) and M (width) and A (height). Before starting the experiment, the researchers pour exactly V cubic units of water into the box. It does not matter if the water barely covers the bottom or if it already reaches the top: the initial volume is always perfectly measured.

The real chaos begins when the interns must place two experimental capsules inside the box. These capsules are tall cylinders that, according to confidential documents, are so high that they exceed any possible water level and stick out well above the box (so the only relevant factor is the circular base they occupy).

Each capsule is defined by its center position on the base (X_i, Y_i) and its diameter D_i. The supervisors ensure that:

- No capsule overlaps another.
- No capsule touches the walls of the box.
- The positions (X_i, Y_i) are always well chosen (although they insist that "perhaps you should double-check them just in case").

The head of the laboratory, famous for speaking in riddles, wrote the evaluation protocol with the following phrases:

1. "The water will always occupy the space left free."
2. "If the water touches the lid, you will know you failed."
3. "The edge separates the careful interns from the careless ones."

Translated: when the capsules are inserted, if the water reaches or exceeds the height of the box, the experiment is considered an overflow.

Your task: Given the box dimensions, the initial water volume, and the data of the two capsules, determine if the water will overflow.

Input

The first line contains four integers N, M, A $(1 \leq N, M, A \leq 10^6)$ and V $(0 \leq V \leq 10^{18})$ representing the length, width, height of the box, and the initial water volume. The total box volume $N \times M \times A$ fits in a 64-bit integer.

The second line contains an integer T (for this problem it will always be $T = 2$).

Each of the following T lines contains three integers X_i, Y_i ($0 < X_i, Y_i < 10^6$) and D_i ($1 \leq D_i \leq 10^6$) where (X_i, Y_i) are the coordinates of the capsule's center and D_i its diameter. The capsule does not touch the walls and capsules do not overlap or touch each other.

It is guaranteed that the data always satisfies the conditions so the capsules fit correctly.

The input must be read from standard input.

Output

Print exactly one line with "Overflow" if the water reaches or exceeds the height of the box or "No overflow" otherwise.

The output must be written to standard output.

Sample input	**Sample output**

`6 5 4 70`
`2`
`2 2 20`
`5 3 2`
	`Overflow`

Sample input	**Sample output**

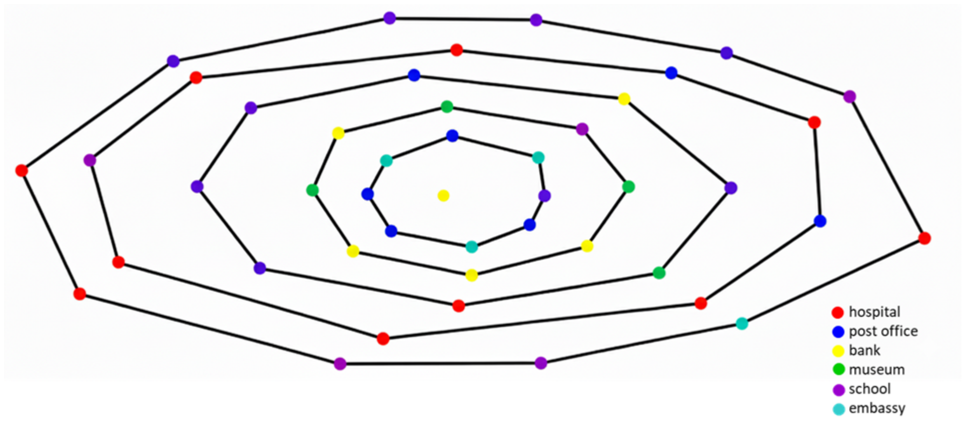
`10 10 10 0`
`2`
`3 3 2`
`7 7 2`
	`No overflow`

Problem G
Urban Planning

Source file name: `urban.cpp`, `urban.py`

The Urban Center for Vehicles (UCV) is planning the expansion bus routes across multiple cities. They have surveyed a list of historical land markers on each city that define concentric "layers" of enclosed zones based on these markers and are interested in finding a ratio between the inner and outer layers to budget the number of buses required for each route.

The outermost zone (Layer 1) must enclose all markers. Subsequent inner zones (Layer 2, Layer 3, etc.) are defined by the remaining markers once the boundary markers of the previous layer are excluded. This process of identifying the territorial boundary and removing its defining markers must be repeated until no markers are left (or too few remain to define a boundary).



Given this layered-zoning protocol, your task is to help the UCV to determine all the concentric territorial zones and obtain the number of markers in both the inner and outer most layers.

Input

The input starts with a line containing an integer $0 < L \leq 10^3$ indicating the number of markers of a city, followed by L lines, each line containing two positive integers X and Y that represent the coordinates of the marker.

The input must be read from standard input.

Output

For the input test case you must print a single line with two integers separated by a single blank space indicating the number of markers of the outer most layer followed by the number of markers of the inner most layer.

The output must be written to standard output.

Sample input

```
10
0 0
0 10
10 10
10 0
4 4
4 8
8 8
8 4
6 6
5 12
```

Sample output

```
5 4
```