TASK	**NOTE**	**DOMINO**	**DOBRA**	**MALI**	**GENIJALAC**	**ALADIN**
source code	`note.pas` `note.c` `note.cpp`	`domino.pas` `domino.c` `domino.cpp`	`dobra.pas` `dobra.c` `dobra.cpp`	`mali.pas` `mali.c` `mali.cpp`	`genijalac.pas` `genijalac.c` `genijalac.cpp`	`aladin.pas` `aladin.c` `aladin.cpp`
input	standard input (*stdin*)					
output	standard output (*stdout*)					
time limit	1 s	1 s	1 s	1 s	5 s	8 s
memory limit	32 MB	32 MB	32 MB	32 MB	32 MB	64 MB
point value	**30**	**50**	**70**	**100**	**120**	**130**
	500					

Author: Marko Ivanković

C major scale consists of 8 tones: c d e f g a h C. For this task we number the notes using numbers 1 through 8. The scale can be played **ascending**, from 1 to 8, **descending**, from 8 to 1, or **mixed**. Write a program that, given the sequence of notes, determines wether the scale was played **ascending**, **descending** or **mixed**.

INPUT

First and only line of input will contain 8 integers, from 1 to 8 inclusive. Each integer will appear exactley once in the input.

OUTPUT

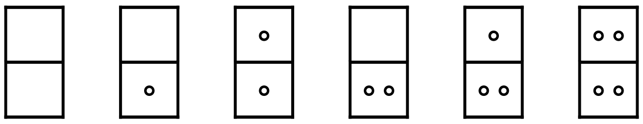
In the first and only line of input print "descending" if the scale was played **descending**, "ascending" if the scale was played **ascending** and "mixed" if the scale was played **mixed**.

SAMPLE TESTS

input	**input**	**input**
`1 2 3 4 5 6 7 8`	`8 7 6 5 4 3 2 1`	`8 1 7 2 6 3 5 4`
output	**output**	**output**
`ascending`	`descending`	`mixed`

Author: Bruno Rahle

Dominoes are gaming pieces used in numerous tile games. Each doimno piece contains two *marks*. Each mark consists of a number of *spots* (possibly zero). The number of spots depends on the *set* size. Each mark in a size **N** domino set can contain between 0 and **N** spots, inclusive. Two tiles are considered identical if their marks have the same number of spots, irregardles of reading order. For example tile with 2 and 8 spot marks is identical to the tile having 8 and 2 spot marks. A proper domino set contains no duplicate tiles. A **complete** set of size **N** contains all posible tiles with **N** or less spots and no duplicate tiles. For example, the complete set of size 2 contains 6 tiles:



Write a program that will determine the total number of spots on all tiles of a complete size **N** set.

INPUT

The first and only line of input contains a single integer, **N** (1 ≤ **N** ≤ 1000), the size of the complete set.

OUTPUT

The first and only line of output should contain a single integer, total number of spots in a complete size **N** set.

SAMPLE TESTS

input 2 **output** 12	**input** 3 **output** 30	**input** 15 **output** 2040

Second sample description:
Size 3 set contains tiles: [0|0], [0|1], [0|2], [0|3], [1|1], [1|2], [1|3], [2|2], [2|3] and [3|3].

Lea runs into a lot of words in her life. A lot of them she finds unpleasant. To compensate for that she started making up plesant words. Lea makes up new words by writing a nice looking string of characters on a piece of paper. She than erases a few of the most nasty looking characters and replaces them with underscores '_'. After that she tries to replace the underscores with more acceptable characters trying to form a plesant word.

Lea considers words plesant if they do not contain 3 sequential vowels, 3 sequential consonants and contain at least one letter 'L'.

In Croatian vowels are letters **A**, **E**, **I**, **O**, **U only**. All other lettters are consonants.

INPUT

The first and only line of input contains a string of characters, at most **100**. The string contains only of **uppercase english letters** and characters '_'. There will be at most **10** characters '_'.

OUTPUT

The first and only line of output should cotain a single integer - the total number of plesant words that can be formed by substituing underscores with uppercase letters of the english alphabet.

Warning: Use 64 bit number formats. *long long* in C/C++, *int64* in Pascal.

SAMPLE TESTS

input L_V **output** 5	**input** V__K **output** 10	**input** JA_BU_K_A **output** 485

Author: Marko Ivanković, Filip Barl

Mirko and Slavko are playing a new game. Again. Slavko starts each round by giving Mirko two numbers A and B, both smaller than 100. Mirko then has to slove the following task for Slavko: how to pair all given A numbers with all given B numbes so that the **maximal sum of such pairs is as small as possible**.

In other words, if during previous rounds Slavko gave numbers a_1, a_2, a_3 .... a_n and b_1, b_2, b_3 ... b_n, determine n pairings (a_i, b_j) such that each number in A sequence is used in exactley one pairing, and each number in B sequenct is used in exactely one pairing and the maximum of all sums $a_i + b_j$ is minimal.

INPUT

The first line of input contains a single integer **N** (1 ≤ **N** ≤ 100000), number of rounds.
Next **N** lines contain two integers **A** and **B** (1 ≤ **A**, **B** ≤ 100), numbers given by Slavko in that round.

OUTPUT

Output consists of **N** lines, one for each round. Each line should contain the smallest maximal sum for that round.

SCORING

Test cases worth 50% of total points have **N** ≤ 200.

SAMPLE TESTS

input 3 2 8 3 1 1 4 **output** 10 10 9	**input** 3 1 1 2 2 3 3 **output** 2 3

Author: Filip Barl, Goran Žužić, Marko Ivanković

Mirko is a genius. But the purpose of his inventions is not always obvious. His latest invention, the Shuffle-o-matic 3175, is one of those. The Shuffle-o-matic is used in a very special way. First Mirko places **N** paper cards, with numbers 1 to **N** printed on them, on the Shuffle-o-matic working surface. Then he inputs the **shuffle sequence** in the special input console and hits the go button. The machine than reads the paper cards and outputs the read sequence of numbers on its output tape. It than shuffles the cards according to the shuffle sequence. After that it reads the newly obtaind sequence and writes it onto a new line on its output tape. Then it procedes to shuffle the cards again **acording to the same shuffle sequence**, scans and writes the output to the tape. The machine does this until it runs out of tape.
After experimenting with the machine Mirko decided to rest a bit on the floor. There he noticed a piece of output tape. The piece is neatly cut just before the **A**-th output row nad just after the **B**-th output row. It is also missing the first **C** number and the last **D** numbers in all rows.
He now wonders how many rows on that piece of paper have the property that **all numbers in the row, that are still on the paper, are in the exact same spot they were before all the shuffling began.**

INPUT

The first line of input contains integers **N**, **A**, **B**, **C** and **D** in that order(1 ≤ N ≤ **500 000**, **A** ≤ **B** ≤ **10^12**, 0 ≤ **C, D** ≤ **N**, **C** + **D** < **N**).
The second line contains the shuffle sequence. The sequence is given as a permutation of numbers 1 to **N**. If the **k**-th number in the shuffle sequence is **x**, after each shuffle the **k**-th element in the resulting sequence is the **x**-th element in the previous sequence.

OUTPUT

In the first and only line of input print the number of rows that have the property Mirko is looking for.

SCORING

Test cases worth 40% total points have **A**, **B**, **C**, **D**, **N** ≤ 2000.

SAMPLE TESTS

input

```
4 1 5 0 1
1 3 4 2
```

output

```
2
```

description

Schuffle-o-matic outputs:

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

Mirko finds:

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

The first and forth row are interesting to Mirko.

input

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

output

```
0
```

description

Schuffle-o-matic outputs:

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

input

```
6 2 11 3 0
6 3 5 4 2 1
```

output

```
1
```

description

Schuffle-o-matic outputs:

```
1 2 3 4 5 6
6 3 5 4 2 1
1 5 2 4 3 6
6 2 3 4 5 1
1 3 5 4 2 6
6 5 2 4 3 1
1 2 3 4 5 6
6 3 5 4 2 1
1 5 2 4 3 6
6 2 3 4 5 1
1 3 5 4 2 6
```

Aladin was walking down the path one day when he found the strangest thing. **N** empty boxes right next to a weird alien machine. After a bit of fumbling around he got the machine to do something. The machine now accepts 4 integers **L**, **R**, **A** and **B**. After that hitting the big red glowing button labeled "NE DIRAJ" causes the machine to go crazy and follow the next routine:

- Set the number of stones in the box labeled **L** to **A** modulo **B**.
- It procedes to fly to the box labeled **L**+1, and set the number of stones there to (2∙**A**) mod **B**.
- It procedes to fly to the box labeled **L**+2, and set the number of stones there to (3∙**A**) mod **B**.
- Generaly, it visits each box labeled between **L** and **R**, and set the number of stones there to ((**X** - **L** + 1)∙**A**) mod **B**. where X is the box label.
- After it visits the box labeled **R**. It settles down for further instructions.

During the game Aladin wonders what is the total number of stones in some range of boxes.

Write a program that simulates the device and answers Aladins questions.

INPUT

The first line contains two integers N i Q (1 ≤ N ≤ 1 000 000 000) (1 ≤ Q ≤ 50 000), number of boxes and number of queries.

The next Q lines contain information about the simulation.

If the line starts with 1, than it follows the format "1 L R A B" (1 ≤ L ≤ R ≤ N) (1 ≤ A, B ≤ 1 000 000), meaning that Aladin keyed in numbers L, R, A and B in the device and allowed the device to do its job.

If the line starts with 2, than it follows the format "2 L R" (1 ≤ L ≤ R ≤ N). Meaning that Aladin wonders how many stones in total are ther stones are in boxes labeled L to R (inclusive).

OUTPUT

For each query beginning with 2 output the answer to that particular query. Queries should be processed in the order they are given in the input.

SCORING

Test cases worth 30% total points have N and Q ≤ 1000.

Test cases worth 70% total points have Q ≤ 1000.

SAMPLE TESTS

input 6 3 2 1 6 1 1 5 1 2 2 1 6 **output** 0 3	**input** 4 5 1 1 4 3 4 2 1 1 2 2 2 2 3 3 2 4 4 **output** 3 2 1 0	**input** 4 4 1 1 4 7 9 2 1 4 1 1 4 1 1 2 1 4 **output** 16 0

First sample description:

The boxes start containing {0, 0, 0, 0, 0, 0}, 0 stones in total.

After that the device sets the stones to {1 mod 2, 2 mod 2, 3 mod 2, 4 mod 2, 5 mod 2, 0} = {1,0,1,0,1,0}, or 3 stones in total.