Albert’s Dream

Albert has a dream: he wants to create his own dictionary (Merriam Webster is his idol and he hopes to have his own name on a thick book one day). Albert realizes that he does not know enough words to create a complete dictionary. But he has many many books and if he could only get all of the unique words out of those, he would be happy.

You are Albert’s best friend and majoring in computer science, so you offer to help with processing all the texts and selecting only the unique words. You are going to write a program that processes text and produces a list of unique words in alphabetical order. Before you do that, Albert gives you his definition of a word. A *word* is a sequence of one or more consecutive alphabetic characters in upper or lower case. Albert also wants your program to be case insensitive, that is words like "APPLE", "apple", "appLE" should be treated as the same.

Input

The input is a text with up to 5,000 lines. Each line has at most 200 characters. The input if terminated by EOF. **Output**

A list of different/unique words that appear in the input text, one per line. You are guaranteed that the number of unique words in the text is no more than 5,000.

Example 1

```
Input:
Albert has a dream: he wants to create his own dictionary (Merriam Webster is
his idol and he hopes to have his own name on a thick book one day).

Output:
a
albert
and
book
create
day
dictionary
dream
has
have
he
his
hopes
idol
is
merriam
name
on
one
own
thick
to
wants
webster
```

Example 2

```
Input:
121()* &* abc *awre@#

Output:
```

```
abc
awre
```

Bit-wise OR

Given 3 non-negative integers N, L, R. Find the **maximum** integer x such that

1. L <= x <= R
2. x **OR** N is maximized (**OR** stands for bit-wise OR operation here), i.e. for all integer y in the close interval [L,R], we have: (y **OR** N) <= (x **OR** N)

Input

There will be 3 integers N, L, R on a single line, where L <= R. You can assume that 0 <= N,L,R <= 4294967295.

Output

Print a line containing the value of x such that x **OR** N is maximized.

Example 1

```
Input:
10 10 10

Output:
10
```

Example 1

```
Input:
100 50 60

Output:
59
```

Efficient Adding

You are tasked with writing a program that adds a sequence of numbers. But the added challenge is to do so efficiently!

The cost of adding two numbers `a` and `b` is equal to their sum `a+b`. For example: to add 1, 2, and 3, you can do it as follows:

1 + 2 = 3, cost of 3
3 + 3 = 6, cost of 6
Total cost = 9

or

2 + 3 = 5, cost of 5
1 + 5 = 6, cost of 6
Total cost = 11

or

1 + 3 = 4, cost of 4
2 + 4 = 6, cost of 6
Total cost = 10

Your goal is to add the numbers so that the cost is as small as possible.

Input

The first line of input contains a positive number `N` (`2 <= N <= 5000`) that tells you how many numbers there are to add.

The second line of input contains those `N` numbers `0 <= n_1, n_2, ..., n_N <= 100,000`.

Output

The minimum total cost of addition followed by a newline.

Example 1

```
Input:
3
1 2 3

Output:
9
```

Example 2

```
Input:
4
1 2 3 4

Output:
19
```

Preparing Problems

Students needs problems to practice techniques they learned in class. Teachers also need problems to test students. You are a problem-setter for an un-named course at NYU.

To avoid plagiarism, problems can not be reused for this course. You will prepare new problems each month.

At the beginning of this year, there are `S` problems that is ready.

Strangely enough, you are able to foresee the number of problems you can create in each of the 12 months of this year as well as the number of problems required in each month.

If `N` problems are required in a month and there are not enough problems at that time, then the quizzes and homeworks for that month are canceled. And the unused problems can be used in the following months.

Write a program to determine if there are enough problems for each month. Please keep in mind that, if a problem is created in month `X` , it can only be used in month `X+1` and the later months.

Input

The first line contains an integer `S` (`0 <= S <= 100`). Number of problems that are ready at the beginning of the year.

The 2-nd line has 12 space separated integers, denoting the number of problems that ca be created each of the 12 months of that year. The months are in the same order as they appear in the calendar. These integers will be between 0 and 20 (inclusive).

The 3-rd line has another 12 space separated integers, the number of problems needed in each of those 12 months (with the same order as above). These integers will be between 0 and 20 (inclusive).

Output

Print 12 lines.

If there are enough problems to meet the requirements in month `i` (`1 <= i <= 12`), print ‘A lot of grading.’ in the `i` -th line, otherwise print ‘No homework.’

Example 1

```
Input:
5
3 0 3 5 8 2 1 0 3 5 6 9
0 0 10 2 6 4 1 0 1 1 2 2

Output:
A lot of grading.
A lot of grading.
No homework.
A lot of grading.
A lot of grading.
A lot of grading.
A lot of grading.
A lot of grading.
A lot of grading.
A lot of grading.
A lot of grading.
A lot of grading.
```

Running Median

Professor Stats has a new theory of how useful the median value could be in describing a real-life data set. She is especially interested in analyzing the running medians. As her research assistant, you are tasked with writing a program that calculates the running median as the data is received, i.e., for each subsequence starting at data point 1 up to data point K (where K ranges from 1 up to the largest index of any data point), you need to compute the median of that subsequence.

If the current sequence has an odd number of elements, then the median is the middle element when the values are in sorted order. For example, for the sequence {1, 3, 6, 2, 7}, the median is 3.

If the current sequence has an even number of elements, then the median is the average of the two middle elements. For example, for the sequence {1, 3, 6, 2, 7, 8}, the median is (3+6)/2 = 4.5. Since Professor Stats lives in the Integer World, you need to report only the integer part of the median, so in this case the program should report 4 (not 4.5).

Input

The input consists of series of integers `X` (`0 <= X <= 2^31`). The total number of integers `N` is less than 100,000. Each line of input contains only one integer, but the numbers may have leading or trailing spaces.

Output

For each input line (each new value), print the current value of the running median.

Example 1

```
Input:
1
3
4
60
70
50
2

Output:
1
2
3
3
4
27
4
```