**Instructions:**

* *Mobile, Pen drive and any types of electrical devices is prohibited to keep aside.*
* *Any kind of printed materials except while papers is prohibited to keep aside.*
* *Staying on the allocated seat is must. In case, there is any technical issues, ask the responsible instructors for any changes of seat.*
* *You must log into google classroom to get and submit the assignment. Any other means will not be allowed in general.*
* *Talking, helping others and any types of idea sharing will be considered as an illegal activities in evaluation time.*
* *In this evaluation, you are allowed to use operators, conditionals, loop and array. Only stdio.h and math.h can be used except some problems.*
* *Marks are distributed as: 60% for case passing, 20% for code correction and 20% for code structure.*

**--------------------------------------------------- A ----------------------------------------------------------**

You will be given an integer n. You need to find the binary equivalence of the integer number.

**Input**

Input consists a line with one integer n. 0 <= n <= 100000000.

**Output**

Print the result in a single line. See the sample

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| **10** | **1010** |
| **513** | **1000000001** |

**--------------------------------------------------- B -----------------------------------------------------------**

In this problem, you have to find the square root of an input number.

**Input**

Input consists of a line containing a single number N.

1<=N<=10000007

**Output**

For each input number N, you have to print the square root of that number up to six decimal points.

**Constraint**

***You are not allowed to use any math.h functions in this problem.***

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| **4** | **2.000000** |
| **3** | **1.732051** |
| **5** | **2.236068** |
| **61** | **7.810250** |

**--------------------------------------------------- C --------------------------------------------------**

In this problem, you will be given a series of characters. You need to print the characters according to the arrangement given in sample cases. The printing shapes are as:

|  |  |  |
| --- | --- | --- |
| **\***  **\*\***  **\*** | **\***  **\*\***  **\*\*\***  **\*\***  **\*** | **\***  **\*\***  **\*\*\***  **\*\*\*\***  **\*\*\***  **\*\***  **\*** |

If an input sequence of characters is lacking of enough characters to make this shape, then the required fields should be covered with an asterisk (\*). Note that you should print as smaller shape as possible.

**Input:**

A series of valid ASCII characters [only alphabets and digits]. It is guaranteed that the total number of input character will not exceed 100000.

**Output:**

Print the characters as the sample. See the sample I/O for clarification.

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| **abcdefghi** | **a**  **bc**  **def**  **gh**  **i** |
| **abcdefghijk** | **a**  **bc**  **def**  **ghij**  **k\*\***  **\*\***  **\*** |

**Explanation of Sample Case 2:**

using **“abcdefghijk”,** there are a lot of possibilities to make the mentioned shape.

But, a 5-lined shape like sample case 1 will overflow the last two characters “jk” which is not allowed.

On the other hand, a 9-lined shape is possible with extra asterisks, but that is not the available smaller shape.

Thus, the 7-lined shape is the preferable output as given in the Sample I/O section.

**--------------------------------------------------- D --------------------------------------------------**

In this problem, you will be given a list of numbers. Then you have to complete some queries. There will be 3 types of queries;

* ***I X***: you need to insert the number X into the array. The condition is that after inserting X into the array, the array should be sorted in ascending order.
* ***D X:*** you need to delete all the occurrences of X from the array. The condition is that after deleting X from the array, the should be sorted in ascending order.
* ***S X:*** you need to find the indices where the element X can be found in the array. If there is no existence of X in the array, just print “not found”.

**Input:**

Input consists of several lines.

First line with two integers N and Q, representing the initial size of the array and the number of queries you need to answer respectively.

Second line consists of N integers in ascending order representing the array.

Next Q lines will describe the queries.

**Output:**

When you are processing the queries for insertion or deletion, you should print the updated array. And at the time of searching only print the indices or not found. See the sample I/O for clarification.

**Constraint:**

***1 <= N, Q <= 1000 and rest of the integers will be a 32-bit integer.***

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| **5 3**  **1 2 3 4 5**  **I 4**  **D 2**  **S 4** | **1 2 3 4 4 5**  **1 3 4 4 5**  **2 3** |
| **5 3**  **1 2 3 4 5**  **I 5**  **D 4**  **S 4** | **1 2 3 4 5 5**  **1 2 3 5 5**  **not found** |

**--------------------------------------------------- E --------------------------------------------------**

We know that every integer can be represented as the multiplication of some prime numbers except 1. Thus, we can have a set of 1 and all the prime factors for an integer n. We can call it a ***Faltu*** set. For example, ***Faltu***(1) = {1}, ***Faltu***(10) = {1, 2, 5}, ***Faltu***(25) = {1, 5} and so on.

In this problem, you will be given a list of positive integers. You need to find the intersection of all the Faltu sets formed for each integers.

**Input**

A line consisting of an integer n (1<=n<=1000) indicating the number of integers. Next line will contain n integers (all the integers will be in the range of [1, 1000000]).

**Output**

Print the resulting set. See the sample I/O for clarification.

|  |  |
| --- | --- |
| **Sample Input** | **Sample Output** |
| **1**  **10** | **1 2 5** |
| **2**  **2 10** | **1 2** |
| **5**  **100 5 35 15 50** | **1 5** |