Question - 1
String to Integer

Given a string, write a function that converts this string into a valid integer. A valid integer is a number preceded optionally by either '+' or '-' and a sequence of digits. The input string can contain non numeric characters as well such as alphabets, symbols, etc. The string which does not contain any numeric character cannot be converted and a 0 should be returned in that case. The string may contain spaces also, which need to be removed before parsing.

Constraints
The integer should be within min and max limits of integer values.
You can define max as 2^{31}-1 and min as -2^{31} .

Samples

Example 1:
Input = " 64"
Output = 64

Example 2:
Input = " -64"
Output = -64

Example 3:
Input = "a-64"
Output = -64

Example 4:
Input = "cbad"
Output = 0

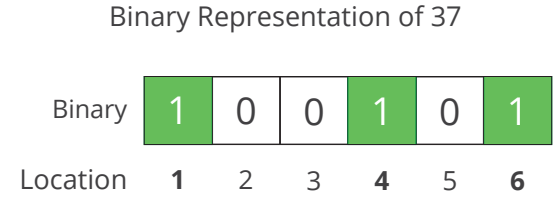
Question - 2
Counting Bits

Given an integer, *n*, determine the following:

1. How many *1*-bits are in its binary representation?
2. The number *n*'s binary representation has *k* significant bits indexed from *1* to *k*. What are the respective positions of each *1*-bit, in ascending order?

For example, the diagram below depicts this information for the value *n = 37*:

/



In the binary representation of *37*, there are three *1*-bits located at the respective 1^{st}, 4^{th}, and 6^{th} positions.

Note: The leftmost *1* bit is always position *1*. Preceding zeros are not considered in determining the position.

Function Description

Complete the function *getOneBits* in the editor below. The function must return a *results* array with the number of *1*'s stored at *results[0]* followed by the positions of all *1*'s in its binary representation in ascending order.

getOneBits has the following parameter(s):

n: an integer

Constraints

- $1 < n < 10^9$

▼ Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The single input is an integer, *n*.

▼ Sample Case 0

Sample Input

```
161
```

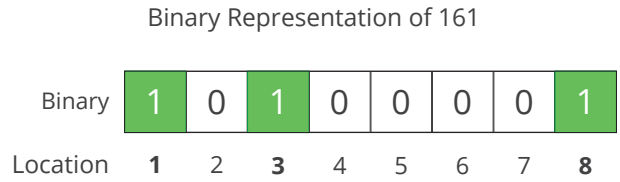
Sample Output

```
3
1
3
8
```

Explanation

The integer $n = (161)_{10}$ converts to $(10100001)_2$:

/



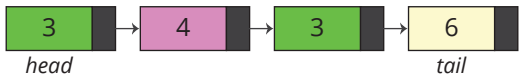
In the binary representation of *161*, there are *3* *1*-bits located at the 1^{st}, 3^{rd}, and 8^{th} positions.

Because there are three *1*-bits, the return array is *3 + 1 = 4* units in length. Store the *1*'s count, *3*, at index *0*. Then store the locations of the *1*-bits in order, low to high. Return the array *[3, 1, 3, 8]* as the answer.

Question - 3
Condensed List

Given a list of integers, remove any nodes that have values that have previously occurred in the list and return a reference to the head of the list.

For example, the following list has a recurrence of the value *3* initially:



Redundant nodes are colored with the same color

Redundant nodes are removed after calling *condense*

Remove the node at position *2* in the list above, *0* based indexing.

Function Description

Complete the function *condense* in the editor below. The function must return a reference to a *LinkedListNode*, the first node of a list that contains only the unique value nodes from the original list, in order.

condense has the following parameter(s):

head: the head of a singly-linked list of integers, a LinkedListNode

Note:. A *LinkedListNode* has two attributes: *data*, an integer, and *next*, a reference to the next item in the list or the language equivalent of *null* at the tail.

/

Constraints

- $1 \le n \le 10^5$
- $0 \le LinkedListNode[i].val \le 1000$

▼ Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer *n*, the size of the array *list*.
Each of the next *n* lines contains an integer *list[i]* where $0 \le i < n$.

▼ Sample Case 0

Sample Input 0

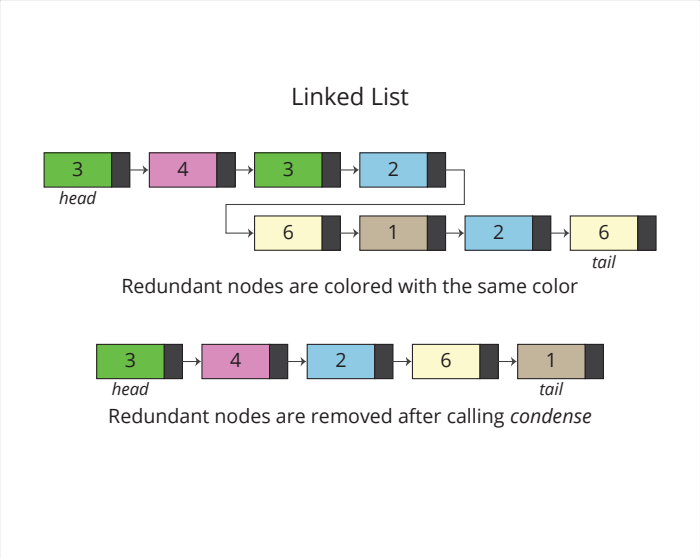
```
STDIN       Function Parameters
-----       -------------------

8       →   list[] Size n = 8
3       →   list[] = [ 3, 4, 3, 2, 6, 1, 2, 6 ]
4
3
2
6
1
2
6
```

Sample Output 0

```
3
4
2
6
1
```

Explanation 0
The list looks like this:



From the first list in the diagram, remove:

- *list[2] = 3*
- *list[6] = 2*
- *list[7] = 6*

Question - 4

/

Roll the String

A single *roll* operation on a string is a circular increment of each character by one. Looking at the English alphabet, characters in the range ascii[a-z], *a* becomes *b*, *b* becomes *c*, and *z* becomes *a*.

Given an array of integers named *roll*, perform a roll operation on the first *roll[i]* characters of *s* for each element *i* in the array. Given a zero indexed string, an operation *roll[i]* affects characters at positions *0* through *(roll[i]-1)*.

Example

s = 'abz'

roll = [3, 2, 1]

Perform the following sequence of operations:

- *roll[0] = 3*: Roll all three characters so '*abz*' becomes '*bca*.'
- *roll[1] = 2*: Roll the first two characters so '*bca*' becomes '*cda*'.
- *roll[2] = 1*: Roll the first character so '*cda*' becomes '*dda*'.

After performing the operations, the final value of *s* is *'dda'*.

Function Description

Complete the function *rollTheString* in the editor below.

rollTheString has the following parameter(s):

string s: the string to operate on

int roll[n]: an array of integers indicating the number of items in *s* to roll

Returns:

string : the resulting string after all roll operations have been performed

Constraints

- Each character in *s* is a character in the range ascii[a-z].
- $1 \le length\ of\ s \le 10^5$
- $1 \le n \le 10^5$
- $1 \le roll[i] \le length\ of\ s$, where $0 \le i < n$.

▼ Input Format Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains a string *s*.

The next line contains an integer *n*, the size of the array *roll*.

The next *n* lines each contain an element, *roll[i]*, where 0 ≤ i < n.

▼ Sample Case 0

Sample Input 0

```
STDIN   Function
-----   --------------------
abz  →  s = 'abz'
1    →  roll[] size n = 1
3    →  roll = [3]
```

/

Sample Output 0

```
bca
```

Explanation 0

Roll forward the first 3 characters in the substring *s[0] … s[2]*, so *abz* becomes *bca*.

▼ Sample Case 1

Sample Input 1

```
STDIN    Function
-----    -------------------
vwxyz →  s = 'vwxyz'
5     →  roll[] size n = 5
1     →  roll = [1, 2, 3, 4, 5]
2
3
4
5
```

Sample Output 1

```
aaaaa
```

Explanation 1

Perform the *n = 5* operations on *s = vwxyz* described in *roll = [1, 2, 3, 4, 5]*:

- *roll[0] = 1*: Roll forward all characters in the substring *s[0] … s[1−1]*, so ***v****wxyz* becomes ***w****wxyz*.
- *roll[1] = 2*: Roll forward all characters in the substring *s[0] … s[2−1]*, so ***ww****xyz* becomes ***xx****xyz*.
- *roll[2] = 3*: Roll forward all characters in the substring *s[0] … s[3−1]*, so ***xxx****yz* becomes ***yyy****yz*.
- *roll[3] = 4*: Roll forward all characters in the substring *s[0] … s[4−1]*, so ***yyyy****z* becomes ***zzzz****z*.
- *roll[4] = 5*: Roll forward all characters in the substring *s[0] … s[5−1]*, so ***zzzzz*** becomes ***aaaaa***.

/