Question - 1
Sample Question

This is a sample question that helps you understand how input-output works on HackerRank. You do not need to write any code, but please read the question and the answer (code) below for a better understanding, before you start solving the remaining questions.

Sample Question
Given two positive integers a and b, your task is to complete the function **sumNum(int a, int b)** to return their sum.

Sample Input
2
3

Sample Output
5

Question - 2
The Chessboard Pattern Problem

Write a function ***chessBoard()*** that returns a chessboard pattern ("B" for black squares, "W" for white squares). The function takes a number N as input, and generates the corresponding board. N will always be > 0.

The below output comes from an input parameter of 5. The top left square will always be white. There will be N lines of output, each line corresponding to one row of the board. All output should be printed to stdout.

Sample Input:
5 // The size of the chess board

Sample output:
W B W B W
B W B W B
W B W B W
B W B W B
W B W B W

Question - 3
Number Permutations

Your challenge is to write a function ***numPermutation()*** that takes in a number ***N*** and returns an array of all possible permutations of the

number given . For example, if the input number is 123, the output should be:

123
132
213
231
312
321

Please Note: It is invalid to write the number 1 as 001.

Question - 4

Letter Sudoku

Your job is to validate Letter Sudoku solutions. While ordinary Sudoku uses the numbers 0..9, Letter Sudoku uses 9 distinct letters from the alphabet. For each board, write a function ***sudoku()*** that takes in a string and corresponding grid, and returns a string: "**Valid**" or "**Invalid**".

The first line of input consists of the 9 letters that form the puzzle without any spaces between the letters. The Letter Sudoku board is then input as 9 lines of 9 letters each, each letter separated from the other by a space character. A board is valid, if and only if, in every row and column, and in every 3x3 sub-board, each letter appears exactly once. It is super-important to adhere to the case in the output string. A sample Valid input is shown below:

MONASTERY // This is the first line of input consisting of the 9 letters that form the puzzle
9 // denotes the number of columns in the sudoku
9 // denotes the number of rows in the sudoku
R M E T O Y **N A S** // These 9 lines are the Sudoku board
N Y O S M A **R E T**
T S A N R E **Y O M**
E R M **O Y S** A T N
S A Y **R T N** O M E
O N T **E A M** S Y R
M O N A S T **E R Y**
A T S Y E R **M N O**
Y E R M N O **T S A**

Question - 5

Number Sequence Puzzle

What is the next number in this sequence: 1, 11, 21, 1211, 111221,….? Before you tear your hair out, here is the solution: In general, each number in the sequence is formed by "reading" the previous number as a "string of digits" – for each run of *n* instances of digit *d*, append *n* followed by digit *d* to the output string. For example, 1211 consists of one 1, one 2, and two 1s. Therefore, the next number in the sequence is 111221. Simple, isn't it?

Your program should take a number as input in function ***nextNumber()*** and return the next number in the sequence based on the above logic.

Sample input:

225

Sample output:

2215

Question - 6

Just a fraction

Your program input will be a fraction. A fraction consists of two integers, the numerator and the denominator.

The input is provided as 2 numbers - num and den - in the function ***justFraction()*** . Return a string as mentioned below.

Compute the reduced form of the fraction, defined as follows: p/q is reduced if and only if gcd(p, q) = 1 and q > 0. If the denominator given is 0, the function should return “Invalid”; otherwise, the function should return a string of the reduced numerator, denominator, and the decimal form rounded to three decimal places, each separated by one space. You may assume that $-2^{31} < p, q < 2^{31}$. Note that either the numerator or denominator could be negative. (GCD – Greatest Common Divisor)

Sample Input

2

4

Sample Output

1 2 0.500

Question - 1

Sample Question

This is a sample question that helps you understand how input-output works on HackerRank. You do not need to write any code, but please read the question and the answer (code) below for a better understanding, before you start solving the remaining questions.

Sample Question

Given two positive integers a and b, your task is to complete the function **sumNum(int a, int b)** to return their sum.

Sample Input

2
3

Sample Output

5

Question - 2

Number Palindrome Detector

A palindrome is something that reads the same forwards and backwards. For example, these numbers are all palindromes:

1001
47274
6
44444

Note that 100 is not a palindrome, since it's lame to start a number with a leading zero (as in 00100).

Your job is to write a function ***isPalindrome()*** that takes in an array of numbers and returns a string array of "Yes" - without the quotes - (yes, it is a palindrome), or "No" (no, it is not a palindrome).

For the example above, the input will be:

4 // There are 4 numbers in this set
1001
47274
6
44444

and the output will be:

Yes
Yes
Yes
Yes

Question - 3

The Eight Queens Puzzle

A classic problem in chess is to place 8 queens on the board in such a manner that no two queens are on the same path. A queen can more up or down, left or right, and diagonally - forwards and backwards - as long as there are no other pieces in its path. Your input will consist of a chessboard with black squares marked as B, white squares marked as W, and squares with queens marked as Q. There will be 8 input lines, each line corresponding to each row of the chessboard, with 8 characters in each row coded as above. A sample input is shown below (Note that the sample input has spacing between the characters to aid readability - the actual input to the program has no spaces between the characters. The queens are also highlighted in bold).

You will be given chess board in the form of 2D-character array in the function ***isValid() .*** Complete the function to return a string - "Valid" (without the quotes - for a valid solution) or "Invalid" (without the quotes - for an invalid solution). Here's a sample input (a valid solution) for which your program needs to print out "Valid".

```
W B W B W Q W B
B W B Q B W B W
W B W B W B Q B
Q W B W B W B W
W B W B W B W Q
B Q B W B W B W
W B W B Q B W B
B W Q W B W B W
```

There are brute force ways to solve this problem, as well as methods and techniques that use ultra-cool data structures and methods. Your final evaluation will be based not only on correctness, but also on the techniques used to solve the problem. For example, think about how your solution would scale for a 1024-square chessboard where 1024 queens would be placed (You don't have to solve this one, but you may be asked about the technique you used in the interview).

Question - 4

Simply Statistics

The arithmetic mean of N numbers is the sum of the numbers, divided by N. The mode of N numbers is the most frequently-occurring number. Your program must output the mean and mode of a set of numbers.

There will be 2 lines of input in each test. The first line will indicate the count of numbers in the test case - the count will always be greater than 0, and the second line will consist of a space-separated list of numbers - as many numbers as indicated in the first line. Note that 0 and negative numbers are valid inputs. Here's a sample input:

```
5 // There will be five numbers in this
test case
1
2
```

7
3
2

The function ***simplyStats()*** provides you an integer array. You have to return a string that contains 2 numbers separated by a space. The first number is the mean of the input numbers and the second number is the mode.

3 2

Question - 5

Word Search Puzzle

You are given a grid of letters, followed by some words. The words can occur anywhere in the grid on a row or a column, forward or backwards. There are no diagonal words, however. The first line of the input consists of a single number which indicates the size of the letter grid. This will be followed by the letter grid itself with each row on a single line of input. Following the letter grid will be a line which indicates the count of the words to search for. This will be followed by the words themselves, one on each one. A sample input is shown below. (Note that in the sample input, there are spaces between the letters to aid readability. The actual test cases will have no spacing between the letters).

3 // Number of columns of the letter grid
3 // Number of rows of the letter grid
C A T // First row of the letter grid
I D O
M O M // Last row of the letter grid
4 // The number of words to search for
CAT
TOM
ADO
MOM

Write a function ***searchWords()*** that returns a string array with as many elements as the number of words to search for. Each array element will consist of "Yes" (without the quotes - the word is present in the grid) or "No" (without the quotes - the word is not present in the grid). It is super-important that you adhere exactly to the case of the letters in the output.

Question - 6

A Problem of Prime Importance

The objective is simple – to list all the prime numbers from a list of input numbers. The first line of input is a number indicating the count of numbers to follow, one per line. Your program should read each subsequent number from standard input, and for each of the n inputs, your program should print “Prime” (without the quotes - indicating whether the number is a prime number) or “Composite” (without the quotes - indicating whether the number is a composite number). It is super-important that the case of the output match the strings above. All numbers will be in the range $1 < n < 10^9$

Complete a function ***primeOrComposite()*** that accepts an array of integers and returns output corresponding to the numbers in the array.

Sample Input

```
8 // 8 integers follow
541
37
113
3
4
2
5
10
```

Sample Output

```
Prime
Prime
Prime
Prime
Composite
Prime
Prime
Composite
```

Question - 1
Multiples of 3 and 5

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.
Write a function `getSum()` to return the sum of all the multiples of 3 or 5 below a given number passed as parameter to your function.

Note: The function stub is already available to you - you only have to add the implementation within the stub.

Question - 2
Sum of two numbers

You are given two lists representing one non-negative number each. The digits of the number represented by the list is stored in reverse order in the list.
Write a function to add the numbers and return it as a list. The arguments to your function will be the aforementioned two lists.

For example,

Input: `[2, 4, 6],[8, 0, 9]`
Output: `[0, 5, 5, 1]`
Explanation: `642 + 908 = 1550`

Question - 3
Matching brackets

Given a string containing only the characters `'('`, `')'`, `'{'`, `'}'`, `'['` and `']'`, find out if the input string is valid.

The input string is valid if:

- Open brackets are closed by the same type of brackets.
- Open brackets are closed in the correct order.

For example,

Input: `"{}"`
Output: `true`

Input: `"{[]}"`
Output: `true`

Input: `"{}[]"`
Output: `true`

Input: `"{[}]"`
Output: `false`

Implement a function `isValid()` that accepts the aforementioned string and returns the total number of brackets when the string is valid and -1 when it is not.

Question - 4
Simplify path

Given an absolute file path (unix / linux style), reduce it.

For example,

Input: `"/root/"`
Output: `"/root"`

Input: `"/x/./y/../z/"`
Output: `"/x/z"`

`.` represents the current directory and `..` represents parent directory.
Consider `"/../"` as `"/"` and `"/x//y/"` as `"/x/y"`.

Write a function `reducePath()` that accepts a path string and return the reduced path.

Question - 5
Valid Palindrome

Given a string, figure out if it is a palindrome considering only alphanumeric characters and ignoring cases.

For example,

Input: `"A man, a plan, a canal: Panama"`
Output: `true`

Input: `"race a car"`
Output: `false`

Write a function `isPalindrome()` that returns the total length of the original input string when it is a palindrome according to the above mentioned rules or -1 when it is not.

Question - 1
Shifting Strings

The following operations on a string are defined:

- *Left Shift*: A single circular rotation of the string in which the first character becomes the last character and all other characters are shifted one index to the left. For example, *abcde* becomes *bcdea* after one left shift and *cdeab* after two left shifts.
- *Right Shift*: A single circular rotation of the string in which the last character becomes the first character and all other characters are shifted one index to the right. For example, *abcde* becomes *eabcd* after one right shift and *deabc* after two right shifts.

Example

s = 'abcdefg'

leftShifts = 2

rightShifts = 4

The string *abcdefg* shifted left by 2 positions is *cdefgab*. The string *cdefgab* shifted right by 4 positions is *fgabcde*, the string to return.

Function Description

Complete the function *getShiftedString* in the editor below.

getShiftedString has the following parameter(s):

string s: the string to shift

int leftShifts: number of shifts left

int rightShifts: number of shifts right

Returns:

string: a string shifted left or right

Constraints

- $1 \leq length\ of\ s \leq 10^5$
- $0 \leq leftShifts, rightShifts \leq 10^9$
- String *s* consists of lowercase English alphabetic letters only, ascii[a-z].

▼ Input Format for Custom Testing

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

The first line contains *s*, the string to shift.

The second line contains an integer, *leftShifts*.

The third line contains an integer, *rightShifts*.

▼ Sample Case 0

/

Sample Input 0

```
STDIN           Function
-----           --------
abcd     →      s = "abcd"
1        →      leftShifts = 1
2        →      rightShifts = 2
```

Sample Output 0

```
dabc
```

Explanation 0

Initially, *s* is *abcd*.

1. *leftShifts = 1 : abcd → bcda*
2. *rightShifts = 2 : bcda → abcd → dabc*

The function then returns *dabc*.

▼ **Sample Case 1**

Sample Input 1

```
STDIN           Function
-----           --------
a        →      s = "a"
0        →      leftShifts = 0
1        →      rightShifts = 1
```

Sample Output 1

```
a
```

Explanation 1

A one character string is unchanged regardless of the number of shifts performed.

Question - 2
Calculate Region

There is a straight line of students of various heights. The students' heights are given in in the form of an array, in the order, they are standing in the line.

Consider the region of a student as the length of the largest subarray that includes that student's position, and in which that student's height is equal to maximum height among all students present in that subarray. Return the sum of the region of all students.

For example-

- *heights = [1, 2, 1]*
- The longest subarray in which the first student's height is equal to maximum height among all other students is [1]; thus, the length of

/

the region of the first student is 1.

- The longest subarray in which the second student's height is equal to maximum height among all other students is [1, 2, 1]; thus, the length of the region of the second student is 3.
- The longest subarray in which the third student's height is equal to maximum height among all other students is [1]; thus, the length of the region of the third student is 1.
- Thus, the sum of the lengths of all regions of all students is *1+3+1 = 5.*

Function Description

Complete the function *calculateTotalRegion* in the editor below. The function must return the desired sum of all regions.

calculateTotalRegion has the following parameter(s):

heights: an array of the heights of students standing in the line

Constraints

- $1 \leq$ length of heights $\leq 10^5$
- $1 \leq heights[i] \leq 10^9$

▼ Input Format For Custom Testing

The first line contains the only integer n that denotes the size of the array *heights*.

The next *n* lines contain one integer that denotes *heights[i]*.

▼ Sample Case 0

Sample Input For Custom Testing

```
3
1
2
1
```

Sample Output

```
5
```

Explanation

The input corresponds to the example given in the statement. The answer is *5* and it is explained in the statement.

▼ Sample Case 1

Sample Input For Custom Testing

```
4
1
1
1
1
```

Sample Output

```
16
```

Explanation

For example :

heights [1,1,1,1]

- The longest subarray in which first student's height is equal to maximum height among all other students is [1, 1, 1, 1], thus the

/

region of the first student is 4.

- The longest subarray in which the second student's height is equal to maximum height among all other students is [1, 1, 1, 1], thus the region of the second student is 4.
- The longest subarray in which the third student's height is equal to maximum height among all other students is [1, 1, 1, 1], thus the region of the third student is 4.
- The longest subarray in which the fourth student's height is equal to maximum height among all other students is [1, 1, 1, 1], thus the region of the fourth student is 4.
- Thus, the sum of the region of all students is *4+4+4+4 = 16*.

/

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

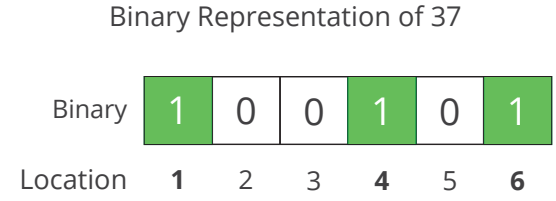
Question - 2
Counting Bits

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
```

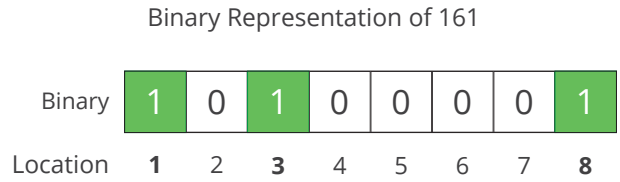
Sample Output

```
3
1
3
8
```

Explanation

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

/



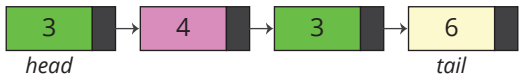
In the binary representation of *161*, there are *3* *1*-bits located at the 1^{st}, 3^{rd}, and 8^{th} positions.

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

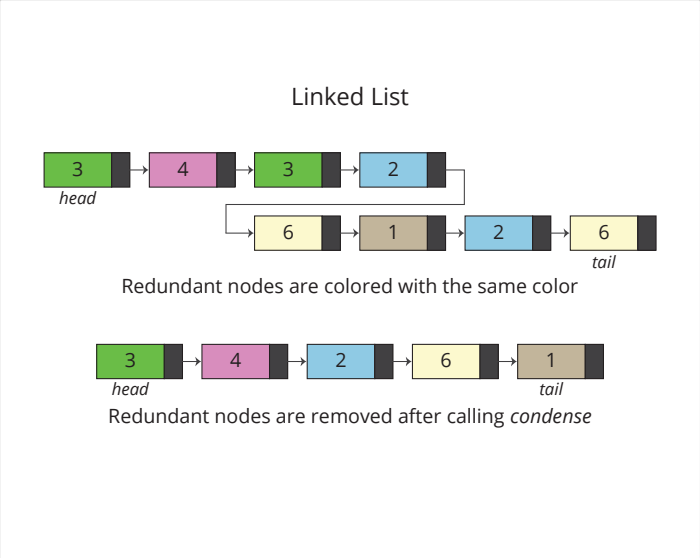
```
STDIN       Function Parameters
-----       -------------------

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***.

/

Question - 1
Maximum among neighbours

Given a list of non negative numbers, find the peak in that list and return the index of the peak. A peak element is an element that is greater than its neighbours. You can assume that no 2 consecutive elements in the array are equal.
If there are multiple peaks in the array, return the index of first peak element.

Example 1:
Input:
[12,23,35,17]
Output: 2

Example 2:
[123,2343,1323,35656,5232,6342,4232]
Output: 1

Question - 2
Compact Array

Given a sorted integer array with no duplicates, compact the array based on continuous range of numbers. If there are no such ranges available, output the list of strings where each element is a string notation of the number.
Input : integer array
Output: List of strings

Example 1
Input : [1,2,3,6,7,8,10,15]
Output :
1 to 3
6 to 8
10
15
Explanation : 1,2,3 form a continuous range and hence is compacted to "1 to 3".Same goes for 6,7,8

Example 2
Input: [10,20,30,40]
Output:
10
20
30
40
Explanation: None of the elements in the array form a continuous range

Question - 3

Roll the String

We define a single *roll* operation on a string to be the circular increment of each character by one. In other words, each character is *rolled* forward and overwritten with the next alphabetic character. 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*, we want to 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)*.

For example, if string *s = abz* and *roll = [3, 2, 1]*, we 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 all the operations, the final value of *s* is *dda*.

Function Description

Complete the function *rollTheString* in the editor below. The function must return the resulting string after all roll operations have been performed.

rollTheString has the following parameter(s):

s: the string to operate on

roll[roll[0],...roll[n-1]]: an array of integers indicating the number of items in *s* to roll

Constraints

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

▼ 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

```
abz
1
3
```

Sample Output 0

```
bca
```

Explanation 0

We want to perform the operation on *s = abz* described in *roll = [3]*. For our first (and only) operation, we roll forward all characters in the substring *s[0] … s[2]* (which ends up being the entire string) so *abz* becomes *bca*.

▼ **Sample Case 1**

Sample Input 1

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

Sample Output 1

```
aaaaa
```

Explanation 1

We want to 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***.

Question - 4

Exencode String

Given a string S, you need to generate an encoded version of this string.
The encoding rules are as follows:
- If you encounter an alphabet in the input string, add it to the output.
- If you encounter a digit 'd', repeat the current output 'd-1' times and append it to the current output.
- If you encounter any character other than alpha numeric, you can ignore it and proceed to the next character.

The digits in the input string can be negative, in which case you will append the current output string once to the output.
If you encounter 0 or 1, do not append anything to the current output.

Example 1:
Input: cisco

Output: cisco

Example 2:
Input: cisco2India
Output: ciscociscoIndia

Example 3:
Input: cisco1India-1Bangalore
Output: ciscoIndiaciscoIndiaBangalore

Question - 1
Lifting Weights

Ollie is new to the gym and is figuring out the maximum weights she can lift. The maximum capacity of the barbell is given as *maxCapacity*. Each barbell plate has a weight, given by *weights[i]*. Now Ollie has to select as many plates as she can but the total weight of the selected plates should not exceed *maxCapacity*. What is the maximum weight of plates Ollie can add to the barbell?
For example, given barbell plates of weights of *1, 3* and *5 lbs* and a barbell of maximum capacity *7 lbs* - the right plates to insert would be *1* and *5 lbs (1+5 = 6),* thus making the right answer *6*.

Function Description

Complete the *weightCapacity* function in the editor below. The function must return an integer denoting the maximum capacity of items that he can purchase.

weightCapacity has two parameters:
weights: An array of *n* integers, where the value of each element *weights[i]* is the weight of each plate *i* (where $0 \le i < n$).
maxCapacity: An integer, the capacity of the barbell.

Constraints

- $1 \le n \le 42$
- $1 \le maxCapacity \le 10^9$
- $1 \le weights[i] \le 10^9$

▼ Input Format For Custom Testing

Locked stub code in the editor reads the following input from stdin and passes it to the function:

The first line contains an integer, *n*, denoting the number of elements in *weights*.
Each line *i* of the *n* subsequent lines contains an integer describing *weights[i]*.
The last line contains an integer, *maxCapacity*, denoting the maximum capacity of the barbell.

▼ Sample Case 0

Sample Input 0

```
STDIN       Function
-----       --------
3         → weights[] size n = 3
1         → weights[] = [ 1, 3, 5 ]
3
5
7         → maxCapacity = 7
```

Help

Sample Output 0

```
6
```

Explanation 0

All the possible combination of items that Ollie can insert are: *{}, {1}, {3}, {5}, {1, 3}, {1, 5}, {3, 5},* and *{1, 3, 5}.*
Out of these combinations, the capacity that can be accommodated is *{1, 5}* making the total weight *6.*

▼ Sample Case 1

Sample Input 1

```
STDIN       Function
-----       --------
4         → weights[] size n = 4
4         → weights[] = [ 4, 8, 5, 9 ]
8
5
9
20        → maxCapacity = 20
```

Sample Output 1

```
18
```

Explanation

All the possible combination of items that Ollie can insert are: *{}, {4}, {8}, {5}, {9}, {4, 8}, {4, 5}, {4, 9}, {8, 5}, {8, 9}, {5, 9}, {4, 8, 5}, {4, 8, 9}, {4, 5, 9}, {8, 5, 9}, {4, 8, 5, 9}.*
Out of these combinations, the capacity that can be accommodated is *{4, 5, 9}* making the total weight *18.*

Question - 2

React: Catalog Viewer

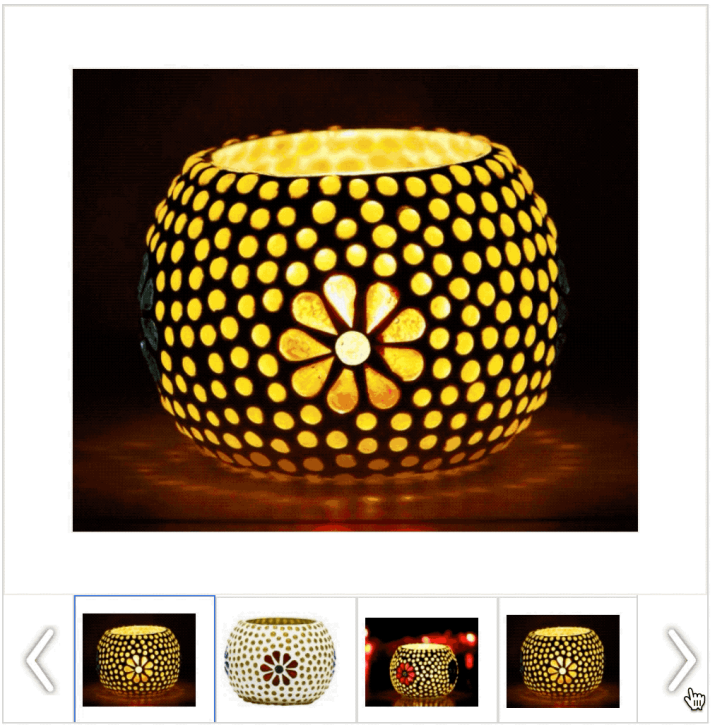
Catalog viewer

Specification Document

Goal:
Implement a simple catalog viewer.

Design:

Catalog Viewer



Implement a catalog viewer for a collection of images.

- The catalog displays the first image when opened.
- Clicking on the *previous* or *next* button displays the previous or next image respectively
- The image list is circular
  - Clicking the *next* button when the last image is showing should display the first image (cycling).
  - Clicking the *previous* button when the first image is showing should display the last image (cycling).

Note: It is not necessary to implement all of the catalog features, only those listed.

Conditions tested:

- Initially, the carousel shows the first image.
- Clicking on any carousel indicator loads the appropriate image in the main view.
- The currently selected thumbnail image should be highlighted.
- There are *previous* and *next* buttons that change the image to be displayed in the carousel.
- Cycling should be allowed.
  - Clicking *next* while showing the last image loads the first image.
  - Clicking *previous* while showing the first image loads the last image.

Question - 3

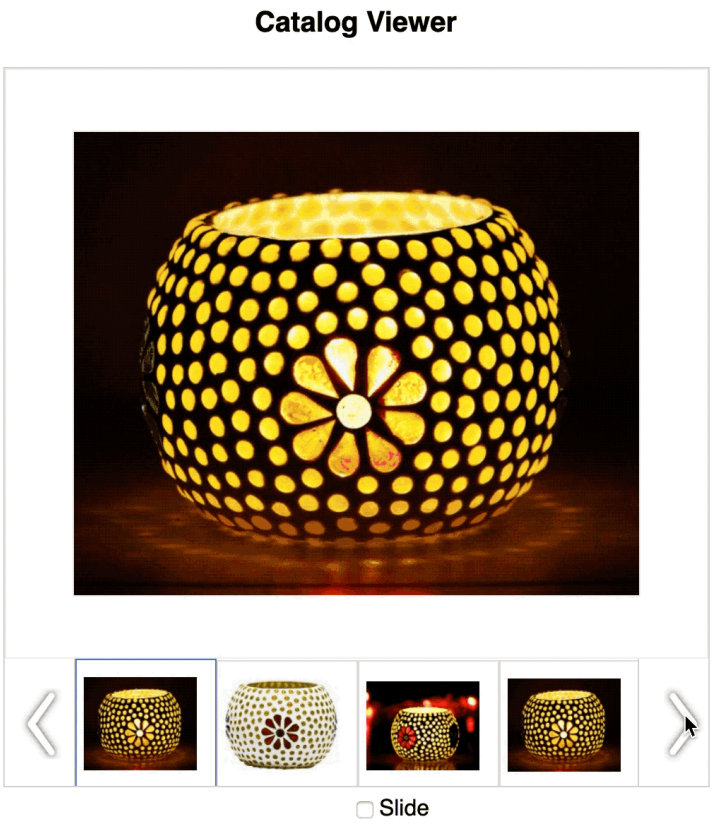
Angular: Catalog Viewer

Catalog viewer

Specification Document

Goal:
Implement a simple catalog viewer.

Design:



Implement a catalog viewer for a collection of images

- The catalog shows the first image when opened.
- Clicking on the *previous* or *next* button displays the previous or next image respectively.
- The image list is circular.
  - Clicking the *next* button when the last image is showing should display the first image (cycling).
  - Clicking the *previous* button when the first image is showing should display the last image (cycling).
- The checkbox with label *Slide* should alternately start and stop the automatic display of images in the carousel. Begin with the currently displayed image and use a 3 second interval.

Note: It is not necessary to implement all of the catalog features, only those listed.

Conditions tested:

- Initially, the carousel shows the first image.
- Clicking on any carousel indicator loads the appropriate image in the main view.
- The currently selected thumbnail image should be highlighted.
- The *previous* and *next* buttons display the appropriate image in the carousel.
- Cycling should be allowed.
    - Clicking *next* while showing the last image loads the first image.
    - Clicking *previous* while showing the first image loads the last image.
- The checkbox labeled “Slide”, when selected, causes cycling through the images with a fixed time interval of 3 seconds.
    - The slide show begins with the current image.
    - Clicking on any carousel indicator or the *previous* or *next* button when the slideshow is active loads the appropriate image and the slideshow continues from that image.

Question - 1

Palindrome Counter

A palindrome is a string that reads the same from the left and from the right. For example, *mom* and *tacocat* are palindromes, as are any single-character strings. Given a string, determine the number of its substrings that are palindromes.

Example

The string is *s = 'tacocat'*. Palindromic substrings are *['t', 'a', 'c', 'o', 'c', 'a', 't', 'coc', 'acoca', 'tacocat']*. There are *10* palindromic substrings.

Function Description

Complete the *countPalindromes* function in the editor.

countPalindromes has the following parameter:

string s: the string to analyze

Returns:

int: an integer that represents the number of palindromic substrings in the given string

Constraints

- $1 \leq |s| \leq 5 \times 10^3$
- *each character of s, s[i] ∈ {'a'-'z'}.*

▼ Input Format For Custom Testing

The first line contains a string, *s*.

▼ Sample Case 0

Sample Input

```
STDIN       Function
-----       -----
aaa     →   s = 'aaa'
```

Sample Output

```
6
```

Explanation

There are *6* possible substrings of *s*: *{'a', 'a', 'a', 'aa', 'aa', 'aaa'}*. All of them are palindromes, so return *6*.

▼ Sample Case 1

Sample Input

Help

```
STDIN       Function
-----       -----
abccba  →   s = 'abccba'
```

Sample Output

```
9
```

Explanation
There are *21* possible substrings of *s*, the following *9* of which are palindromes: *{'a', 'a', 'b', 'b', 'c', 'c', 'cc', 'bccb', 'abccba'}*.

▼ Sample Case 2

Sample Input

```
STDIN     Function
-----     -----
daata  →  s = 'daata'
```

Sample Output

```
7
```

Explanation
There are *15* possible substrings of *s*, the following *7* of which are palindromes: *{'a', 'a', 'a', 'aa', 'ata', 'd', 't'}*.

Question - 2
Lifting Weights

Ollie is new to the gym and is figuring out the maximum weights she can lift. The maximum capacity of the barbell is given as *maxCapacity*. Each barbell plate has a weight, given by *weights[i]*. Now Ollie has to select as many plates as she can but the total weight of the selected plates should not exceed *maxCapacity*. What is the maximum weight of plates Ollie can add to the barbell?
For example, given barbell plates of weights of *1, 3* and *5 lbs* and a barbell of maximum capacity *7 lbs* - the right plates to insert would be *1* and *5 lbs (1+5 = 6),* thus making the right answer *6*.

Function Description

Complete the *weightCapacity* function in the editor below. The function must return an integer denoting the maximum capacity of items that he can purchase.

weightCapacity has two parameters:
weights: An array of *n* integers, where the value of each element *weights[i]* is the weight of each plate *i* (where *0 ≤ i < n*).
maxCapacity: An integer, the capacity of the barbell.

Constraints

- *1 ≤ n ≤ 42*

- $1 \leq maxCapacity \leq 10^9$
- $1 \leq weights[i] \leq 10^9$

▼ Input Format For Custom Testing

Locked stub code in the editor reads the following input from stdin and passes it to the function:

The first line contains an integer, *n*, denoting the number of elements in *weights*.
Each line *i* of the *n* subsequent lines contains an integer describing *weights[i]*.
The last line contains an integer, *maxCapacity*, denoting the maximum capacity of the barbell.

▼ Sample Case 0

Sample Input 0

```
STDIN       Function
-----       --------
3         → weights[] size n = 3
1         → weights[] = [ 1, 3, 5 ]
3
5
7         → maxCapacity = 7
```

Sample Output 0

```
6
```

Explanation 0
All the possible combination of items that Ollie can insert are: *{}, {1}, {3}, {5}, {1, 3}, {1, 5}, {3, 5},* and *{1, 3, 5}.*
Out of these combinations, the capacity that can be accommodated is *{1, 5}* making the total weight *6.*

▼ Sample Case 1

Sample Input 1

```
STDIN       Function
-----       --------
4         → weights[] size n = 4
4         → weights[] = [ 4, 8, 5, 9 ]
8
5
9
20        → maxCapacity = 20
```

Sample Output 1

```
18
```

Explanation
All the possible combination of items that Ollie can insert are: *{}, {4}, {8}, {5}, {9}, {4, 8}, {4, 5}, {4, 9}, {8, 5}, {8, 9}, {5, 9}, {4, 8, 5}, {4, 8, 9}, {4, 5, 9}, {8, 5, 9}, {4, 8, 5, 9}.*
Out of these combinations, the capacity that can be accommodated is *{4, 5, 9}* making the total weight *18.*