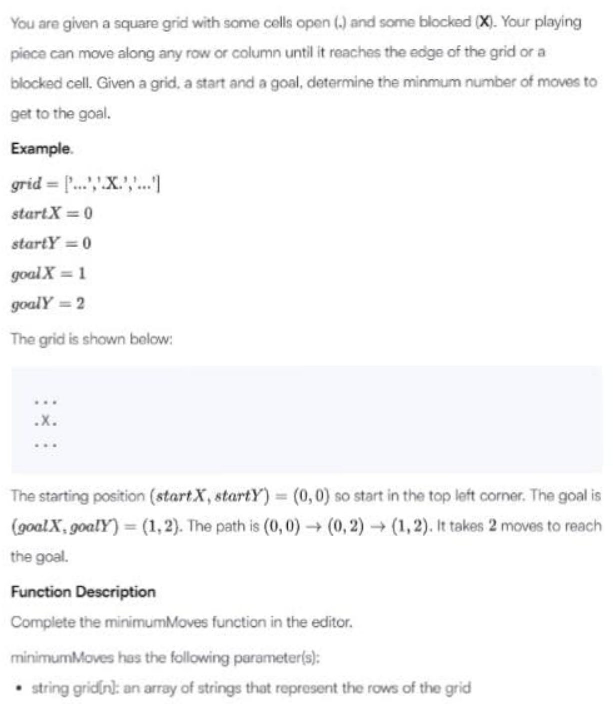


You are given a square grid with some cells open (.) and some blocked (**X**). Your playing piece can move along any row or column until it reaches the edge of the grid or a blocked cell. Given a grid, a start and a goal, determine the minmum number of moves to get to the goal.

Example.

$grid = [\text{'...','.X.','...'}]$

$startX = 0$

$startY = 0$

$goalX = 1$

$goalY = 2$

The grid is shown below:

```
...
.X.
...
```

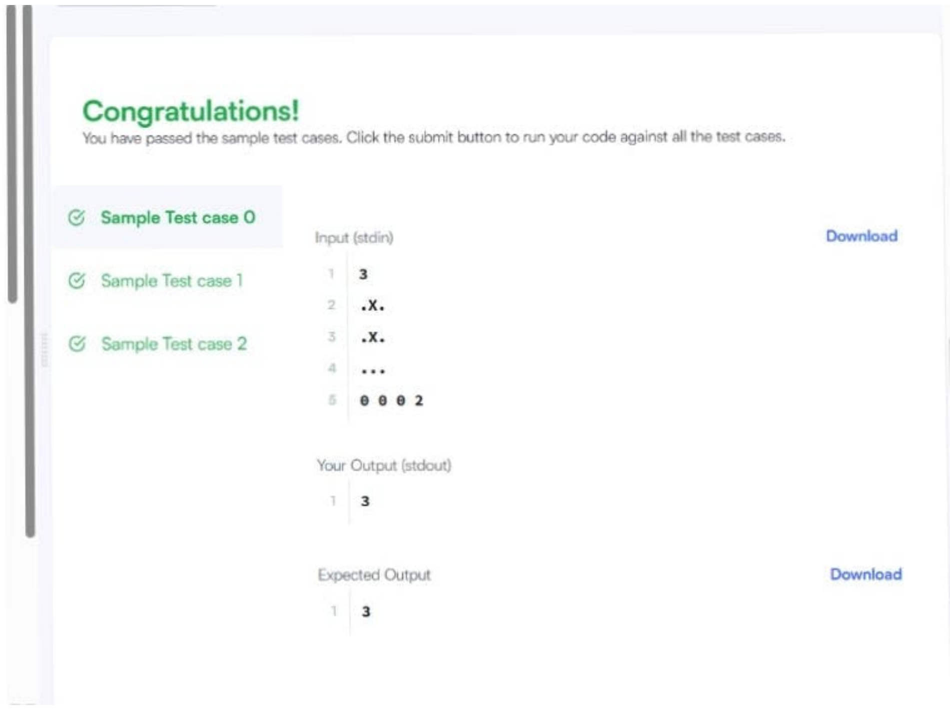
The starting position $(startX, startY) = (0, 0)$ so start in the top left corner. The goal is $(goalX, goalY) = (1, 2)$. The path is $(0, 0) \rightarrow (0, 2) \rightarrow (1, 2)$. It takes 2 moves to reach the goal.

Function Description

Complete the minimumMoves function in the editor.

minimumMoves has the following parameter(s):

- string grid[n]: an array of strings that represent the rows of the grid



Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Input (stdin)

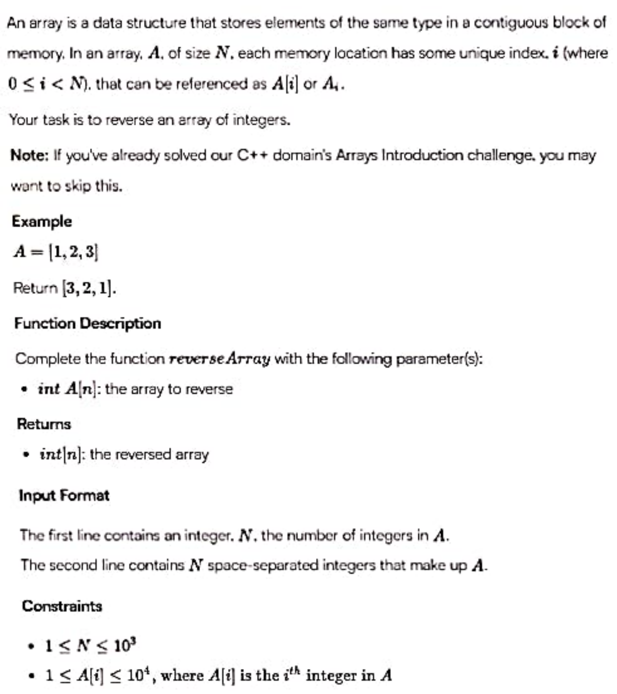
```
3
.X.
.X.
...
0 0 0 2
```

Your Output (stdout)

```
3
```

Expected Output

```
3
```



An array is a data structure that stores elements of the same type in a contiguous block of memory. In an array, A, of size N, each memory location has some unique index, i (where $0 \leq i < N$), that can be referenced as $A[i]$ or A_i.

Your task is to reverse an array of integers.

Note: If you've already solved our C++ domain's Arrays Introduction challenge, you may want to skip this.

Example

$A = [1, 2, 3]$

Return $[3, 2, 1]$.

Function Description

Complete the function $reverseArray$ with the following parameter(s):

- $int\ A[n]$: the array to reverse

Returns

- $int[n]$: the reversed array

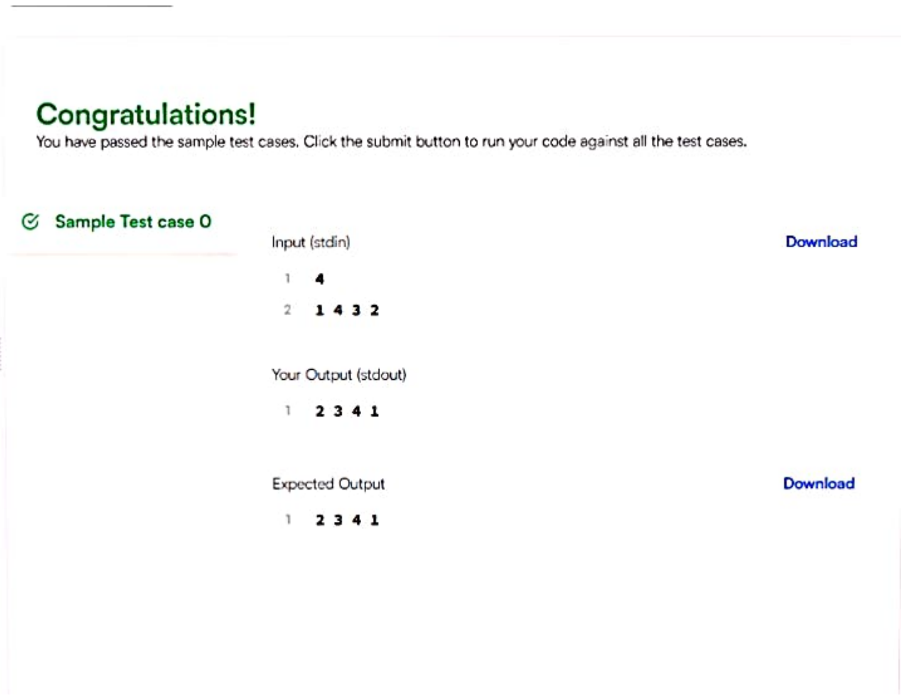
Input Format

The first line contains an integer, N, the number of integers in A.

The second line contains N space-separated integers that make up A.

Constraints

- $1 \leq N \leq 10^3$
- $1 \leq A[i] \leq 10^4$, where $A[i]$ is the i^{th} integer in A



Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

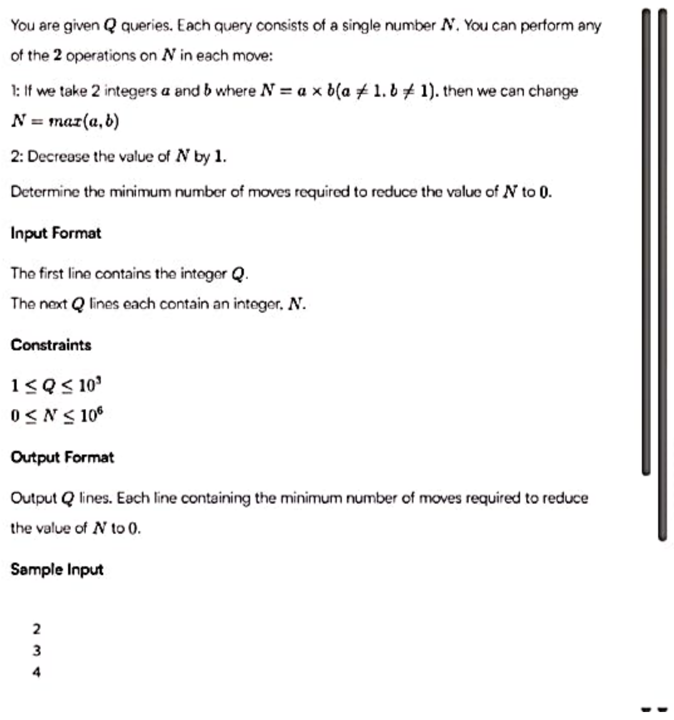
```
4
1 4 3 2
```

Your Output (stdout)

```
2 3 4 1
```

Expected Output Download

```
2 3 4 1
```



You are given Q queries. Each query consists of a single number N. You can perform any of the $\mathbf{2}$ operations on N in each move:

1: If we take 2 integers a and b where $N = a \times b (a \neq 1, b \neq 1)$, then we can change $N = max(a, b)$

2: Decrease the value of N by $\mathbf{1}$.

Determine the minimum number of moves required to reduce the value of N to $\mathbf{0}$.

Input Format

The first line contains the integer Q.
The next Q lines each contain an integer, N.

Constraints

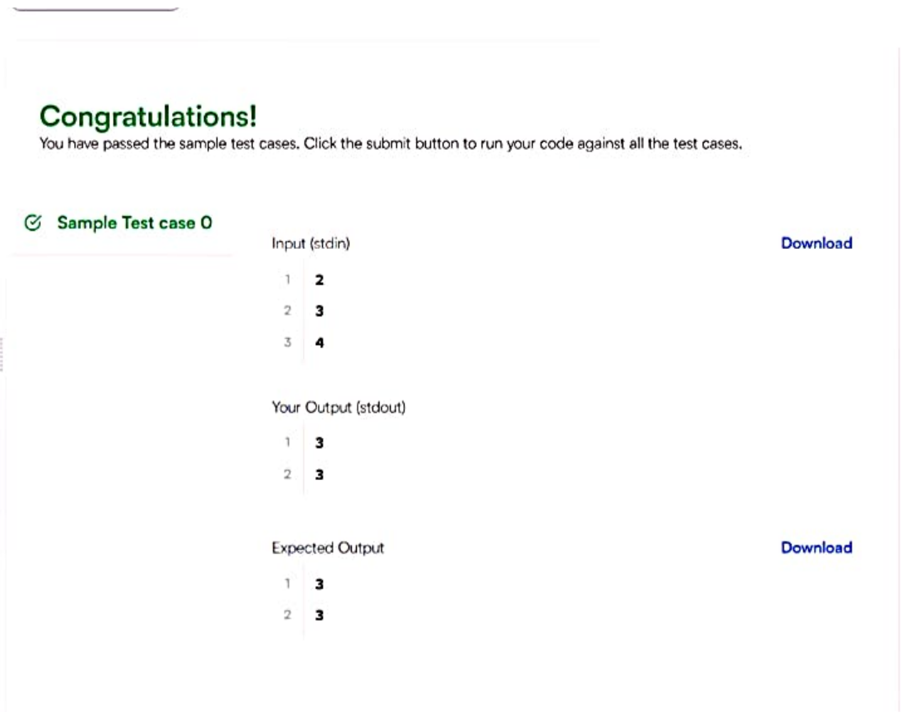
$1 \leq Q \leq 10^3$
$0 \leq N \leq 10^6$

Output Format

Output Q lines. Each line containing the minimum number of moves required to reduce the value of N to 0.

Sample Input

```
2
3
4
```



Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

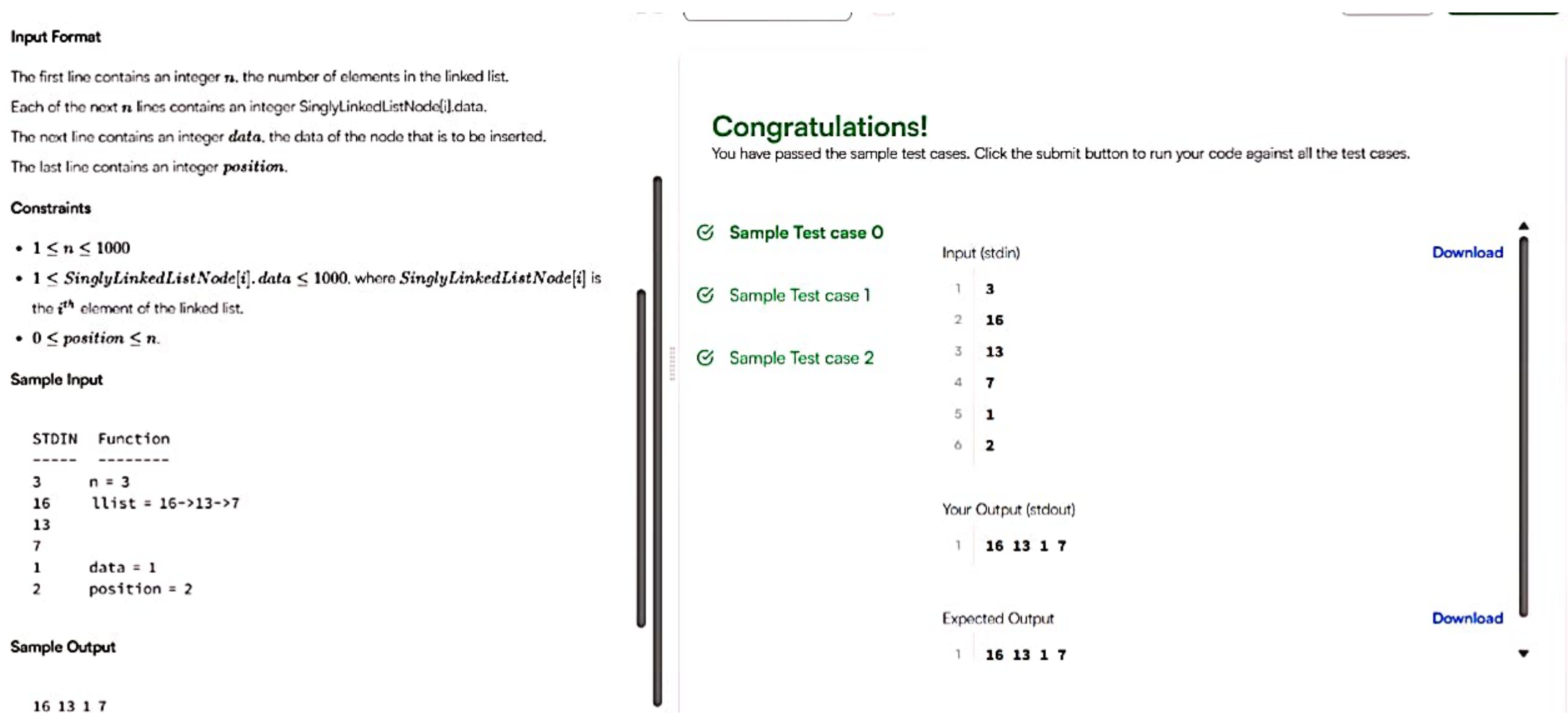
```
2
3
4
```

Your Output (stdout)

```
3
3
```

Expected Output Download

```
3
3
```



Input Format

The first line contains an integer n, the number of elements in the linked list.

Each of the next n lines contains an integer SinglyLinkedListNode[i].data.

The next line contains an integer $data$, the data of the node that is to be inserted.

The last line contains an integer $position$.

Constraints

- $1 \leq n \leq 1000$
- $1 \leq SinglyLinkedListNode[i].data \leq 1000$, where $SinglyLinkedListNode[i]$ is the i^{th} element of the linked list.
- $0 \leq position \leq n$.

Sample Input

```
STDIN   Function
-----   --------
3       n = 3
16      llist = 16->13->7
13
7
1       data = 1
2       position = 2
```

Sample Output

```
16 13 1 7
```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Input (stdin)

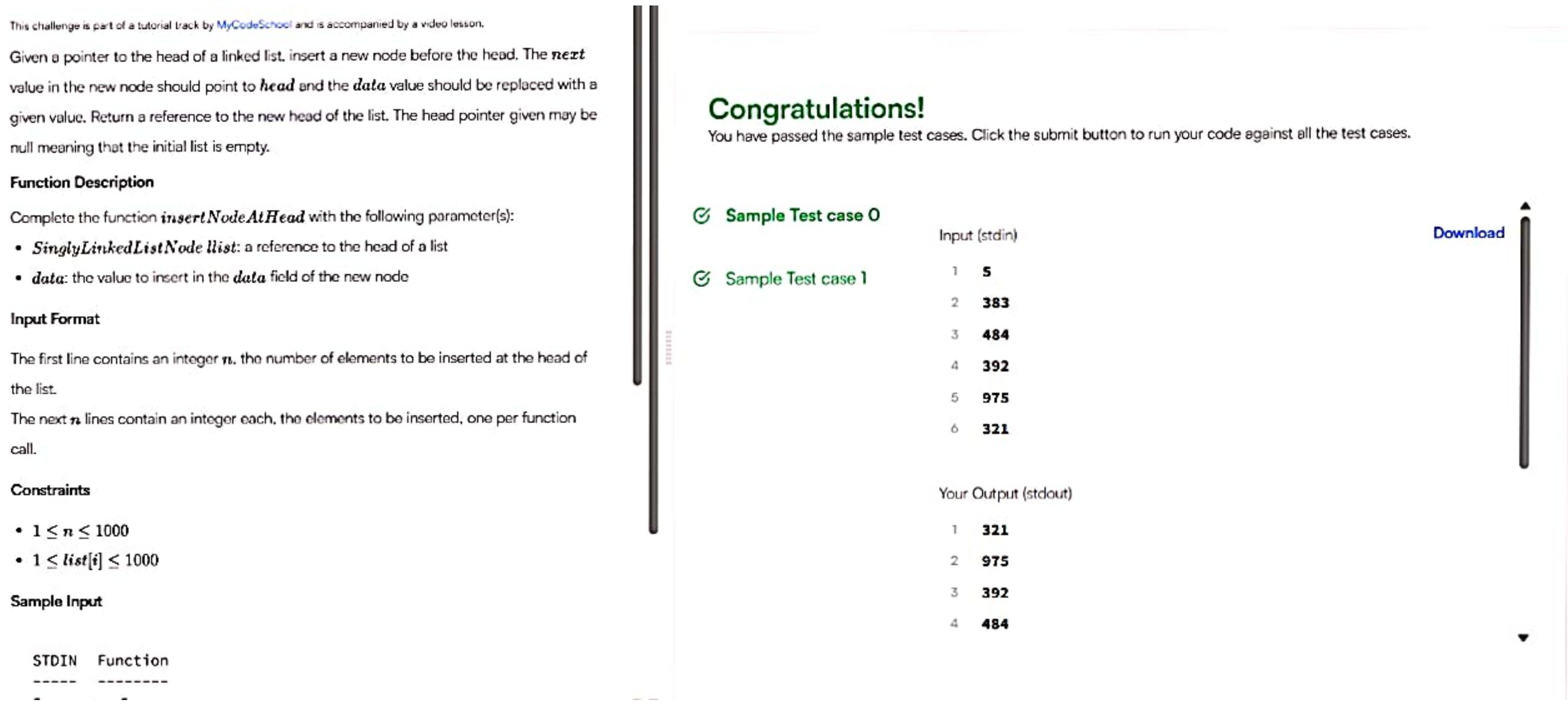
```
3
16
13
7
1
2
```

Your Output (stdout)

```
16 13 1 7
```

Expected Output

```
16 13 1 7
```



This challenge is part of a tutorial track by MyCodeSchool and is accompanied by a video lesson.

Given a pointer to the head of a linked list, insert a new node before the head. The $next$ value in the new node should point to $head$ and the $data$ value should be replaced with a given value. Return a reference to the new head of the list. The head pointer given may be null meaning that the initial list is empty.

Function Description

Complete the function $insertNodeAtHead$ with the following parameter(s):

- $SinglyLinkedListNode\ llist$: a reference to the head of a list
- $data$: the value to insert in the $data$ field of the new node

Input Format

The first line contains an integer n, the number of elements to be inserted at the head of the list.

The next n lines contain an integer each, the elements to be inserted, one per function call.

Constraints

- $1 \le n \le 1000$
- $1 \le list[i] \le 1000$

Sample Input

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

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Sample Test case 1

Input (stdin)

Download

```
5
383
484
392
975
321
```

Your Output (stdout)

```
321
975
392
484
```