

1. Dynamic Array

The screenshot shows the HackerRank interface for the 'Dynamic Array' challenge. The left sidebar contains navigation links: Submissions, Leaderboard, Discussions, and Editorial. The main content area is divided into two columns. The left column contains the problem description, which includes instructions to declare a 2D array `arr` and an integer `lastAnswer`, and to process two types of queries. The right column displays a 'Congratulations!' message, indicating that the user has passed the sample test cases. Below the message, there is a 'Sample Test case 0' section with input and output data.

Problem Description:

- Declare a 2-dimensional array, `arr`, with `n` empty arrays, all zero-indexed.
- Declare an integer, `lastAnswer`, and initialize it to 0.

You need to process two types of queries:

- Query: `1 x y`
 - Compute `idx = (x ⊕ lastAnswer)`.
 - Append the integer `y` to `arr[idx]`.
- Query: `2 x y`
 - Compute `idx = (x ⊕ lastAnswer)`.
 - Set `lastAnswer = arr[idx][y % size(arr[idx])]`.
 - Store the new value of `lastAnswer` in an answers array.

Notes:

- `⊕` is the bitwise XOR operation, which corresponds to the `^` operator in most languages. Learn more about it on [Wikipedia](#).
- `%` is the modulo operator.
- Finally, `size(arr[idx])` is the number of elements in `arr[idx]`.

Function Description

Complete the `dynamicArray` function with the following parameters:

- `int n`: the number of empty arrays to initialize in `arr`
- `int queries[q][3]`: 2-D array of integers

Returns

- `int[]`: the results of each type 2 query in the order they are presented

Sample Test case 0

Input (stdin)

```
1 2 5
2 1 0 5
3 1 1 7
4 1 0 3
5 2 1 0
6 2 1 1
```

Your Output (stdout)

```
7
3
```

Expected Output

```
7
3
```

2. Print the Elements of Linked List

This challenge is part of a [MyCodeSchool](#) tutorial track and is accompanied by a [video lesson](#).

This exercise focuses on traversing a linked list. You are given a pointer to the **head** node of a linked list. The task is to print the **data** of each node, one per line. If the head pointer is **null**, indicating the list is empty, nothing should be printed.

Function Description

Complete the `printLinkedList` function with the following parameter(s):

- `SinglyLinkedListNode head`: a reference to the head of the list

Print

- For each node, print its **data** value on a new line (console.log in Javascript).

Input Format

The first line of input contains **n**, the number of elements in the linked list. The next **n** lines contain one element each, the **data** values for each node.

Note: Do not read any input from stdin/console. Complete the `printLinkedList` function in the editor below.

Constraints

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

Sample Input

```
3
16
13
```

Sample Test case 0

Input (stdin)

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

Your Output (stdout)

```
1 16
2 13
```

Expected Output

```
1 16
2 13
```

Congratulations!

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

3. Insert a node at a Tail of a LinkedList

This challenge is part of a tutorial track by [MyCodeSchool](#) and is accompanied by a video lesson.

You are given the pointer to the head node of a linked list and an integer to add to the list. Create a new node with the given integer. Insert this node at the tail of the linked list and return the head node of the linked list formed after inserting this new node. The given head pointer may be null, meaning that the initial list is empty.

Function Description

Complete the `insertNodeAtTail` function with the following parameters:

- `SinglyLinkedListNode pointer head`: a reference to the head of a list
- `int data`: the data value for the node to insert

Returns

- `SinglyLinkedListNode pointer`: reference to the head of the modified linked list

Input Format

The first line contains an integer **n**, the number of elements in the linked list. The next **n** lines contain an integer each, the value that needs to be inserted at tail.

Constraints

- $1 \leq n \leq 1000$
- $1 \leq list_i \leq 1000$

Sample Input

```
STDIN      Function
3          insertNodeAtTail(head, 5)
16          16
13          13
5           5
```

Test case 0

Compiler Message

Success

Input (stdin)

```
1 5
2 141
3 302
4 164
5 530
6 474
```

Expected Output

```
1 141
```

Congratulations

You solved this challenge. Would you like to challenge your friends? [Facebook](#) [Twitter](#) [LinkedIn](#) [Next Challenge](#)

4. Tree: Level Order Traversal

HackerRank | Prepare | Data Structures | Trees | Tree: Level Order Traversal | Exit Full Screen View

Problem

Given a pointer to the root of a binary tree, you need to print the level order traversal of this tree. In level-order traversal, nodes are visited level by level from left to right. Complete the function `levelOrder` and print the values in a single line separated by a space.

For example:

```
graph TD
    1 --> 2
    1 --> 5
    2 --> 3
    2 --> 6
    5 --> 4
```

For the above tree, the level order traversal is 1 → 2 → 5 → 3 → 6 → 4.

Input Format

You are given a function,

```
void levelOrder(Node * root) {
}
```

Submissions

Leaderboard

Constrains

24°C Cloudy

Search

ENG IN 2001 23-11-2025

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)

```
1 6
2 1 2 5 3 6 4
```

Download

Your Output (stdout)

```
1 1 2 5 3 6 4
```

Expected Output

```
1 1 2 5 3 6 4
```

Download

5. Reverse a doubly linked list

HackerRank | Prepare | Data Structures | Linked Lists | Reverse a doubly linked list | Exit Full Screen View

Problem

This challenge is part of a tutorial track by MyCodeSchool

Given the pointer to the head node of a doubly linked list, reverse the order of the nodes in place. That is, change the next and prev pointers of the nodes so that the direction of the list is reversed. Return a reference to the head node of the reversed list.

Note: The head node might be NULL to indicate that the list is empty.

Function Description

Complete the reverse function in the editor below.

reverse has the following parameter(s):

- DoublyLinkedListNode head: a reference to the head of a DoublyLinkedList

Returns

- DoublyLinkedListNode: a reference to the head of the reversed list

Input Format

The first line contains an integer t , the number of test cases.

Each test case is of the following format:

- The first line contains an integer n , the number of elements in the linked list.
- The next n lines contain an integer each denoting an element of the linked list.

Constraints

- $1 \leq t \leq 10$
- $0 \leq n \leq 1000$
- $0 \leq \text{DoublyLinkedListNode.data} \leq 1000$

Submissions

Leaderboard

24°C Cloudy

Search

ENG IN 2009 23-11-2025

Change Theme Language C++11

```
77 * };
78 *
79 */
80
81 DoublyLinkedListNode* reverse(DoublyLinkedListNode* llist) {
82     if (llist == nullptr) return llist;
83     DoublyLinkedListNode* current = llist;
84     DoublyLinkedListNode* temp = nullptr;
85     while (current != nullptr) {
86         temp = current->prev;
87         current->prev = current->next;
88         current->next = temp;
89         current = current->prev;
90     }
91     if (temp != nullptr) {
92         llist = temp->prev;
93     }
94     return llist;
95 }
96
97 int main()
98 {
99     ofstream fout(getenv("OUTPUT_PATH"));
```

Line: 99 Col: 42

Upload Code as File

Test against custom input

Run Code

Submit Code

6. Tree: Preorder Traversal

The screenshot shows the HackerRank interface for the 'Tree: Preorder Traversal' problem. The left sidebar contains links for Problem, Submissions, Leaderboard, Discussions, and Editorial. The main content area on the left provides the problem description: 'Complete the `preOrder` function in the editor below, which has 1 parameter: a pointer to the root of a binary tree. It must print the values in the tree's preorder traversal as a single line of space-separated values.' It also lists constraints ($1 \leq \text{Nodes in the tree} \leq 500$), the output format, and a sample input tree diagram. The tree diagram is a binary tree with root 1, left child 2, right child 5, 2's left child 3, 3's right child 4, and 5's right child 6. The 'Sample Output' is '1 2 5 3 4 6'. The right panel shows a 'Congratulations!' message and test case results. Two test cases are passed: 'Sample Test case 0' and 'Sample Test case 1'. Both show input '6' and output '1 2 5 3 4 6'. The bottom status bar shows the system time as 2013 and 23-11-2025.

7. Tree: Preorder Traversal

The screenshot shows the HackerRank interface for the 'Tree: Inorder Traversal' problem. The left sidebar contains links for Submissions, Leaderboard, Discussions, and Editorial. The main content area on the left provides the problem description: 'Print the tree's inorder traversal as a single line of space-separated values.' It lists constraints ($1 \leq \text{Nodes in the tree} \leq 500$), the output format, and the same sample input tree diagram as in the previous screenshot. The 'Sample Output' is '1 2 3 4 5 6'. The 'Explanation' states: 'The tree's inorder traversal results in 1 2 3 4 5 6 as the required result.' The right panel shows a 'Congratulations!' message and test case results. Two test cases are passed: 'Sample Test case 0' and 'Sample Test case 1'. Both show input '6' and output '1 2 3 4 5 6'. The bottom status bar shows the system time as 2019 and 23-11-2025.

8. Simple Text Editor

The screenshot shows the HackerRank interface for the 'Simple Text Editor' problem. The left sidebar contains links for Problem, Submissions, Leaderboard, Discussions, and Editorial. The main content area displays the problem description, sample input/output, and an explanation. The right sidebar shows the 'Congratulations!' message and a list of sample test cases.

Problem

```
1 xy
2 3 2
3 4
4 3 1
```

Sample Output

```
c
y
a
```

Explanation

Initially, S is empty. The following sequence of 8 operations are described below:

1. $S = ""$. We append abc to S , so $S = "abc"$.
2. Print the 3rd character on a new line. Currently, the 3rd character is c .
3. Delete the last 3 characters in S (abc), so $S = ""$.
4. Append xy to S , so $S = "xy"$.
5. Print the 2nd character on a new line. Currently, the 2nd character is y .
6. Undo the last update to S , making S empty again (i.e., $S = ""$).
7. Undo the next to last update to S (the deletion of the last 3 characters), making $S = "abc"$.
8. Print the 1st character on a new line. Currently, the 1st character is a .

Sample Test case 0

```
1 8
2 1 abc
3 3 3
4 2 3
5 1 xy
6 3 2
7 4
8 4
9 3 1
```

Your Output (stdout)

```
1 c
2 y
3 a
```

9. Down to Zero (2)

The screenshot shows the HackerRank interface for the 'Down to Zero II' problem. The left sidebar contains links for Submission, Leaderboard, Discussions, and Editorial. The main content area displays the problem constraints, output format, sample input/output, and an explanation. The right sidebar shows the 'Congratulations!' message and a list of sample test cases.

Submission

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

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
```

Sample Output

```
3
3
```

Explanation

For test case 1, We only have one option that gives the minimum number of moves. Follow $3 \rightarrow 2 \rightarrow 1 \rightarrow 0$. Hence, 3 moves.

For the case 2, we can either go $4 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 0$ or $4 \rightarrow 2 \rightarrow 1 \rightarrow 0$. The 2nd option is more optimal. Hence, 3 moves.

Sample Test case 0

Input (stdin)

```
1 2
2 3
3 4
```

Your Output (stdout)

```
1 3
2 3
```

Expected Output

```
1 3
2 3
```

10. Jesse and Cookies

The screenshot shows the HackerRank interface for the 'Jesse and Cookies' problem. The left sidebar contains links for 'Problem', 'Submissions', 'Leaderboard', and 'Discussions'. The main content area displays the problem description, an example, and the function description. On the right, a 'Congratulations!' message is shown, indicating that the user has passed the sample test cases. Below the message, there are sections for 'Sample Test case 0', 'Input (stdin)', 'Your Output (stdout)', and 'Expected Output', each with a 'Download' button.

Problem

Jesse loves cookies and wants the sweetness of some cookies to be greater than value k . To do this, two cookies with the least sweetness are repeatedly mixed. This creates a special combined cookie with:

sweetness = $(1 \times \text{Least sweet cookie} + 2 \times \text{2nd least sweet cookie})$.

This occurs until all the cookies have a sweetness $\geq k$.

Given the sweetness of a number of cookies, determine the minimum number of operations required. If it is not possible, return -1 .

Example

$k = 9$

$A = [2, 7, 3, 6, 4, 6]$

The smallest values are 2, 3.

Remove them then return $2 + 2 \times 3 = 8$ to the array. Now $A = [8, 7, 6, 4, 6]$.

Remove 4, 6 and return $4 + 6 \times 2 = 16$ to the array. Now $A = [16, 8, 7, 6]$.

Remove 6, 7, return $6 + 2 \times 7 = 20$ and $A = [20, 16, 8, 7]$.

Finally, remove 8, 7 and return $7 + 2 \times 8 = 23$ to A . Now $A = [23, 20, 16]$.

All values are $\geq k = 9$ so the process stops after 4 iterations. Return 4.

Function Description

Complete the cookies function in the editor below.

cookies has the following parameters:

- $\text{int } k$: the threshold value
- $\text{int } A[n]$: an array of sweetness values

Submissions

Leaderboard

Discussions

Editorial

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)

```
1 6 7
2 1 2 3 9 10 12
```

Download

Your Output (stdout)

```
1 2
```

Expected Output

```
1 2
```

Download

11. Left Rotation

The screenshot shows the HackerRank interface for the 'Left Rotation' problem. The left sidebar contains links for 'Submissions', 'Leaderboard', 'Discussions', and 'Editorial'. The main content area displays the problem description, constraints, sample input, sample output, and explanation. On the right, a 'Congratulations!' message is shown, indicating that the user has passed the sample test cases. Below the message, there are sections for 'Sample Test case 0', 'Input (stdin)', 'Your Output (stdout)', and 'Expected Output', each with a 'Download' button.

Submissions

Leaderboard

Discussions

Editorial

integers, and d , the number of left rotations to perform.

The second line contains n space-separated integers that describe $arr[]$.

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq d \leq n$
- $1 \leq arr[i] \leq 10^6$

Sample Input

STDIN	Function
5 4	$n = 5$ $d = 4$
1 2 3 4 5	$arr = [1, 2, 3, 4, 5]$

Sample Output

```
5 1 2 3 4
```

Explanation

To perform $d = 4$ left rotations, the array undergoes the following sequence of changes:

$[1, 2, 3, 4, 5] \rightarrow [2, 3, 4, 5, 1] \rightarrow [3, 4, 5, 1, 2] \rightarrow [4, 5, 1, 2, 3] \rightarrow [5, 1, 2, 3, 4]$

Submissions

Leaderboard

Discussions

Editorial

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)

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

Download

Your Output (stdout)

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

Expected Output

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

Download

12. Insert a node at a specific position in a linked list

The screenshot shows the HackerRank interface for the challenge "Insert a node at a specific position in a linked list". The left sidebar contains navigation links: Submission, Leaderboard, Discussions, and Editorial. The main content area is divided into two sections. The left section contains the problem description, an example, and the function description. The right section displays the test cases and the user's output.

Problem Description:

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

Given a pointer to the head node of a linked list and an integer to insert at a certain position, create a new node with the given integer as its *data* attribute, insert this node at the desired position, and return the head node.

A position of 0 indicates the head, a position of 1 indicates one node away from the head, and so on. The head pointer given may be null, meaning that the initial list is empty.

Example

head refers to the first node in the list $1 \rightarrow 2 \rightarrow 3$

data = 4

position = 2

Insert a node at position 2 with *data* = 4. The new list is $1 \rightarrow 2 \rightarrow 4 \rightarrow 3$

Function Description

Complete the function `insertNodeAtPosition` with the following parameters:

- `SinglyLinkedListNode pointer list`: a reference to the head of the list
- `data`: an integer value to insert as data in the new node
- `position`: an integer position to insert the new node, zero-based indexing

Returns

- `SinglyLinkedListNode pointer`: a reference to the head of the revised list

Input Format

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

Test Cases:

- Sample Test case 0
- Sample Test case 1
- Sample Test case 2

Input (stdin):

```
1 3
2 16
3 13
4 7
5 1
6 2
```

Your Output (stdout):

```
1 16 13 1 7
```

Expected Output:

```
1 16 13 1 7
```

The bottom of the image shows a Windows taskbar with the date 2042-11-23 and time 20:42.