

## Data Structures and Algorithms

Name: Lithika V

Reg.No: 11249M026

### 1. Dynamic Array

The screenshot shows a challenge titled "Dynamic Array" on the HackerRank platform. The challenge involves processing two types of queries on a 2-dimensional array. The challenge details, input, and output sections are visible.

**Challenge Details:**

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

You need to process two types of queries:

- Query:  $1 \ x \ y$ 
  - Compute  $\text{idz} = (x \oplus \text{lastAnswer})$ .
  - Append the integer  $y$  to  $\text{arr}[\text{idz}]$ .
- Query:  $2 \ x \ y$ 
  - Compute  $\text{idz} = (x \oplus \text{lastAnswer})$ .
  - Set  $\text{lastAnswer} = \text{arr}[\text{idz}][y \% \text{size}(\text{arr}[\text{idz}])]$ .
  - Store the new value of  $\text{lastAnswer}$  in an answers array.

**Notes:**

- $\oplus$  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[idz])` is the number of elements in  $\text{arr}[\text{idz}]$ .

**Function Description:**

Complete the `dynamicArray` function with the following parameters:

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

**Returns:**

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

**Input (stdin)**

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

**Your Output (stdout)**

```
1 7
2 3
```

**Expected Output**

```
7
```

**Download**

**Editorial**

**Light rain**  
At night

ENG IN 23-11-2025

## 2. Print the Elements of Linked List

This screenshot shows the 'Print the Elements of a Linked List' challenge on the HackerRank platform. The challenge is part of a MyCodeSchool tutorial track. The problem description states that the task is to print the `data` value of each node in a linked list, one per line. If the head pointer is `null`, nothing should be printed. The function signature is `printLinkedList(SinglyLinkedListNode head)`. The input format specifies that the first line contains `n`, the number of elements, followed by `n` lines of `data` values. The expected output is a single line of `n` `data` values separated by newlines. Sample test cases show that for `n=2` with inputs `2` and `16`, the output should be `16` on a new line. The challenge has been completed successfully.

## 3. Insert a node at a Tail of a LinkedList

This screenshot shows the 'Insert a Node at the Tail of a Linked List' challenge on the HackerRank platform. The challenge is part of a tutorial track by MyCodeSchool. The problem description states that given a pointer to the head of a linked list and an integer `n`, the task is to insert a new node at the tail of the list and return the head of the modified list. The given head pointer may be `null`, indicating an empty list. The function signature is `insertNodeAtTail(SinglyLinkedListNode pointer, int data)`. The input format specifies that the first line contains `n`, the number of elements, followed by `n` lines of `data` values. The expected output is the head of the modified linked list. Sample test cases show that for `n=6` with inputs `5`, `141`, `302`, `164`, `538`, and `474`, the output should be `141` on a new line. The challenge has been completed successfully.

#### 4. Tree: Level Order Traversal

The screenshot shows a solved challenge on the HackerRank platform. The challenge title is "Tree: Level Order Traversal". The problem statement asks to print the level order traversal of a binary tree. It provides a sample tree diagram and an example input-output pair. The solution code is provided as follows:

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

The challenge interface includes tabs for "Problem", "Submissions", "Leaderboard", and "Discussions". On the right, there's a "Congratulations!" message, sample test cases, and download links for input and output files.

#### 5. Reverse a doubly linked list

The screenshot shows a solved challenge on the HackerRank platform. The challenge title is "Reverse a doubly linked list". The problem statement asks to reverse the order of nodes in a doubly linked list. It provides a note about the head node being NULL, a function description, and input/output formats. The solution code is provided as follows:

```
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"));
```

The challenge interface includes tabs for "Problem", "Submissions", "Leaderboard", and "Discussions". On the right, there are buttons for "Upload Code as File", "Test against custom input", "Run Code", and "Submit Code".

## 6. Tree: Preorder Traversal

Tree: Preorder Traversal | Hackerrank Dynamic array solution

hackerrank.com/challenges/tree-preorder-traversal/problem?isFullScreen=true

HackerRank | Prepare Data Structures Trees Tree: Preorder Traversal

Problem

Complete the `preOrder` function in the editor below, which has 1 parameter: a pointer to the root node of a binary tree. It must print the values in the tree's preorder traversal as a single line of space-separated values.

Input Format

Our test code passes the root node of a binary tree to the `preOrder` function.

Constraints

1 ≤ Nodes in the tree ≤ 500

Output Format

Print the tree's preorder traversal as a single line of space-separated values.

Sample Input

```
1
 \
 2
   \
   5
   / \
  3   6
   \
   4
```

Sample Output

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

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
```

Download

Sample Test case 1

Input (stdin)

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

Your Output (stdout)

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

Expected Output

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

Download

2013 ENG IN 23-11-2025

## 7. Tree: Inorder Traversal

Tree: Inorder Traversal | Hackerrank Dynamic array solution

hackerrank.com/challenges/tree-inorder-traversal/problem?isFullScreen=true

HackerRank | Prepare Data Structures Trees Tree: Inorder Traversal

Submission

Constraints

1 ≤ Nodes in the tree ≤ 500

Output Format

Print the tree's inorder traversal as a single line of space-separated values.

Sample Input

```
1
 \
 2
   \
   5
   / \
  3   6
   \
   4
```

Sample Output

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

Explanation

The tree's inorder traversal results in 1 2 3 4 5 6 as the required result.

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
```

Download

Sample Test case 1

Input (stdin)

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

Your Output (stdout)

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

Expected Output

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

Download

2019 ENG IN 23-11-2025

## 8. Simple Text Editor

The screenshot shows a challenge titled "Simple Text Editor" on the HackerRank platform. The challenge interface includes a sidebar with navigation links like Problem, Submissions, Leaderboard, and Discussions. The main area displays sample input and output, an explanation of the operations performed on a string  $S$ , and a "Congratulations!" message indicating success. The Windows taskbar at the bottom shows the date as 23-11-2023 and the time as 20:22.

Sample Input:

```
1 xy
3 2
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 **a**bc to  $S$ , so  $S = "abc"$ .
2. Print the 3<sup>rd</sup> character on a new line. Currently, the 3<sup>rd</sup> character is c.
3. Delete the last 3 characters in  $S$  (**a**bc), so  $S = ""$ .
4. Append **xy** to  $S$ , so  $S = "xy"$ .
5. Print the 2<sup>nd</sup> character on a new line. Currently, the 2<sup>nd</sup> 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 1<sup>st</sup> character on a new line. Currently, the 1<sup>st</sup> character is 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:

```
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
```

Input (stdin):

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

Download

24°C Light rain

Search

ENG IN

20:22

23-11-2023

## 9. Down to Zero (2)

The screenshot shows a challenge titled "Down to Zero II" on the HackerRank platform. The challenge interface includes a sidebar with navigation links like Submission, Leaderboard, and Editorial. The main area displays constraints, an output format section, sample input and output, an explanation of the moves required to reduce a value to zero, and a "Congratulations!" message indicating success. The Windows taskbar at the bottom shows the date as 23-11-2023 and the time as 20:28.

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
```

Sample Output:

```
3
3
```

Explanation:

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

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

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:

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

Input (stdin):

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

Download

Your Output (stdout):

```
1 3
2 3
```

Expected Output:

```
1 3
2 3
```

Download

24°C Light rain

Search

ENG IN

20:28

23-11-2023

## 10. Jesse and Cookies

The screenshot shows the 'Jesse and Cookies' challenge on the HackerRank platform. The challenge details state that 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:  $\text{sweetness} = (1 \times \text{Least sweet cookie} + 2 \times \text{2nd least sweet cookie})$ . 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:

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

The challenge interface includes a sidebar with 'Problem', 'Submissions', 'Leaderboard', 'Discussions', and 'Editorial'. The status bar at the bottom shows '24°C Light rain', 'ENG IN', '20:30', and the date '23-11-2025'.

## 11. Left Rotation

The screenshot shows the 'Left Rotation' challenge on the HackerRank platform. The challenge details state that you are given an array  $arr$  of  $n$  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]

The challenge interface includes a sidebar with 'Problem', 'Submission', 'Leaderboard', 'Discussions', and 'Editorial'. The status bar at the bottom shows 'Rain warning In effect', 'ENG IN', '20:33', and the date '23-11-2025'.

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

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.

Sample Test case 0

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
---	----	----	---	---

Download

Download

24°C  
Light rain

Search

ENG IN

2042 23-11-2025