

1)

The image displays three separate screenshots of a web-based programming environment, likely LeetCode or a similar platform, showing different test results for the same C++ code snippet.

Screenshot 1 (Left): Shows the code accepted with runtime 0 ms. Input: nums = [2,7,11,15], target = 9. Output: [0,1]. Expected: [0,1].

Screenshot 2 (Middle): Shows the code accepted with runtime 0 ms. Input: nums = [3,2,4], target = 6. Output: [1,2]. Expected: [1,2].

Screenshot 3 (Right): Shows the code accepted with runtime 0 ms. Input: nums = [3,3], target = 6. Output: [0,1]. Expected: [0,1].

In all three cases, the code is identical:

```
10 result[i] = j;
11     *returnSize = 2;
12     return result;
13 }
14 }
15 }
16 }
17 *returnSize = 0;
18 return malloc(sizeof(int) * 0);
19 }
```

This screenshot shows a single instance of a programming environment displaying a successful test result for the same C++ code snippet.

Input: nums = [3,3], target = 6. Output: [0,1]. Expected: [0,1].

The code is identical to the ones in the previous screenshots:

```
10 result[i] = j;
11     *returnSize = 2;
12     return result;
13 }
14 }
15 }
16 }
17 *returnSize = 0;
18 return malloc(sizeof(int) * 0);
19 }
```

26)

The image shows two side-by-side screenshots of the LeetCode platform's code editor interface. Both windows display the same C++ code for a problem, which finds the first missing positive integer in an array.

Code Editor (Left Window):

```
4 int k = 1;
5 for (int i = 1; i < numSize; i++) {
6     if (num[i] != num[i - 1]) {
7         num[k] = num[i];
8         k++;
9     }
10 }
11
12 return k;
13 }
```

Code Editor (Right Window):

```
4 int k = 1;
5 for (int i = 1; i < numSize; i++) {
6     if (num[i] != num[i - 1]) {
7         num[k] = num[i];
8         k++;
9     }
10 }
11
12 return k;
13 }
```

User Interface Elements:

- Header:** Shows the user's profile picture and name "Sannidhi Patil". It also indicates a "Premium subscription".
- Code Area:** Displays the C++ code for both windows.
- Testcase Tab:** Both windows have a "Testcase" tab open.
- Test Result Tab:** Both windows have a "Test Result" tab open.
- Input:** Both windows show the input array as [1,1,2].
- Output:** Both windows show the output as [1,2].
- Expected:** Both windows show the expected output as [1,2].
- Footer:** Includes links for "Try New Features", "Orders", "My Playgrounds", "Settings", "Appearance", and "Sign Out". It also features a "Contribute a testcase" button and an "Activate Windows" link.

35)

The image displays three separate screenshots of a programming environment, likely LeetCode, showing different submissions for the same problem. Each screenshot includes a code editor, a test result summary, and a user profile sidebar.

Screenshot 1: Shows the following code:

```
if (target > nums[j] && target < nums[j + 1]) {  
    return (j + 1);  
}  
else {  
    j++;  
}  
}  
}  
return 0;  
}
```

Screenshot 2: Shows the following code:

```
if (target > nums[j] && target < nums[j + 1]) {  
    return (j + 1);  
}  
else {  
    j++;  
}  
}  
}  
return 0;  
}
```

Screenshot 3: Shows the following code:

```
if (target > nums[j] && target < nums[j + 1]) {  
    return (j + 1);  
}  
else {  
    j++;  
}  
}  
}  
return 0;  
}
```

In all three cases, the code is accepted with a runtime of 0 ms. The user profile sidebar shows the user's name as Sannidhi Patil and indicates they have access to premium features.

1971)

The screenshot shows two separate code submissions in an IDE, likely LeetCode, for a problem involving graph traversal.

Submission 1 (Top):

- Code:**

```
C </> Code
C < Auto
27     } return true;
28
29     for (int i = 0; i < adjCount[node]; i++) {
30         int neighbor = adjList[node][i];
```
- Testcase:** Accepted, Runtime: 0 ms
- Input:**

```
n =
3
```
- edges:**

```
[{0,1},{1,2},{2,0}]
```
- source:**

```
0
```
- destination:**

```
2
```
- Output:**

```
true
```
- Expected:**

```
true
```

Submission 2 (Bottom):

- Code:**

```
C </> Code
C < Auto
27     } return true;
28
29     for (int i = 0; i < adjCount[node]; i++) {
30         int neighbor = adjList[node][i];
```
- Testcase:** Accepted, Runtime: 0 ms
- Input:**

```
n =
6
```
- edges:**

```
[{0,1},{0,2},{3,5},{5,4},{4,3}]
```
- source:**

```
0
```
- destination:**

```
5
```
- Output:**

```
false
```
- Expected:**

```
false
```

A user profile sidebar is visible on the right, showing "Sannidhi Patil" and "Access all features with our Premium subscription!". It includes links for "My Lists", "Notebook", "Progress", "Points", "Try New Features", "Orders", "My Playgrounds", "Settings", "Appearance", and "Sign Out". A "Activate Windows" watermark is present at the bottom of both panes.

203)

The image shows four separate browser windows, each representing a different test case for the LeetCode problem "Remove Linked List Elements".

- Testcase 1:** Input: head = [1,2,3,4,5,6], val = 6. Output: [1,2,3,4,5]. Expected: [1,2,3,4,5]. Status: Accepted.
- Testcase 2:** Input: head = [1,2,3,4,5], val = 1. Output: []. Expected: []. Status: Accepted.
- Testcase 3:** Input: head = [7,7,7,7], val = 7. Output: []. Expected: []. Status: Accepted.
- Testcase 4:** Input: head = [1], val = 1. Output: []. Expected: []. Status: Accepted.

In all four cases, the code successfully removes the specified elements from the linked list. The LeetCode interface shows the user's profile (Sannidhi Patil) and various navigation links like My Lists, Notebook, Progress, Points, etc.

```
Accepted C++ Auto
6 */;
7 struct ListNode* removeElements(struct ListNode* head, int val) {
8     struct ListNode dummy;
9     dummy.next = head;
10    struct ListNode* curr = &dummy;
11    ...
12    while(curr->next != NULL){
13        ...
14    }
15 }
```

```
Accepted C++ Auto
6 */
7 struct ListNode* removeElements(struct ListNode* head, int val) {
8     struct ListNode dummy;
9     dummy.next = head;
10    struct ListNode* curr = &dummy;
11    ...
12    while(curr->next != NULL){
13        ...
14    }
15 }
```

```
Accepted C++ Auto
6 */
7 struct ListNode* removeElements(struct ListNode* head, int val) {
8     struct ListNode dummy;
9     dummy.next = head;
10    struct ListNode* curr = &dummy;
11    ...
12    while(curr->next != NULL){
13        ...
14    }
15 }
```

```
Accepted C++ Auto
6 */
7 struct ListNode* removeElements(struct ListNode* head, int val) {
8     struct ListNode dummy;
9     dummy.next = head;
10    struct ListNode* curr = &dummy;
11    ...
12    while(curr->next != NULL){
13        ...
14    }
15 }
```

148)

The image shows three separate code editor windows side-by-side, each with a different input and output. All three windows have the same header: "Saved" at the top left, "Sannidhi Patil" with a profile picture at the top right, and a sidebar with "My Lists", "Notebook", and "Progress" buttons.

Left Window (Input: [4,2,1,3], Output: [1,2,3,4]):

```
C:\v Auto
40 struct ListNode* sortList(ListNode* head) {
41     if (!head || !head->next) return head;
42     struct ListNode* left = head;
43     struct ListNode* right = head->next;
44     struct ListNode* ans = sortList(left);
45     struct ListNode* ans2 = sortList(right);
46     return mergeTwoLists(ans, ans2);
47 }
```

Middle Window (Input: [-1,5,3,4,0], Output: [-1,0,3,4,5]):

```
C:\v Auto
40 struct ListNode* sortList(ListNode* head) {
41     if (!head || !head->next) return head;
42     struct ListNode* left = head;
43     struct ListNode* right = head->next;
44     struct ListNode* ans = sortList(left);
45     struct ListNode* ans2 = sortList(right);
46     return mergeTwoLists(ans, ans2);
47 }
```

Right Window (Input: [], Output: []):

```
C:\v Auto
40 struct ListNode* sortList(ListNode* head) {
41     if (!head || !head->next) return head;
42     struct ListNode* left = head;
43     struct ListNode* right = head->next;
44     struct ListNode* ans = sortList(left);
45     struct ListNode* ans2 = sortList(right);
46     return mergeTwoLists(ans, ans2);
47 }
```

Each window has a "Testcase" tab selected, showing "Accepted" status and runtime of 0 ms. The sidebar on the right of each window also includes "Points" and "Contribute a testcase" buttons.

206)

The screenshot shows two versions of the same C++ code for reversing a singly linked list. The code uses three pointers: curr, next, and prev. It iterates through the list, updating the pointers to reverse the links. Both versions are marked as "Accepted" with a runtime of 0 ms.

```
Code Accepted X
C v Auto
14     next = curr->next;
15     curr->next = prev;
16     prev = curr;
17     curr = next;
18 }
19
20 return prev;
21 }
```

User's Solution:

```
Code Accepted X
C v Auto
14     next = curr->next;
15     curr->next = prev;
16     prev = curr;
17     curr = next;
18 }
19
20 return prev;
21 }
```

Test Result:

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input: head = [1,2,3,4,5]
Output: [5,4,3,2,1]
Expected: [5,4,3,2,1]

Contribute a testcase

Reference Solution:

```
Code Accepted X
C v Auto
14     next = curr->next;
15     curr->next = prev;
16     prev = curr;
17     curr = next;
18 }
19
20 return prev;
21 }
```

Test Result:

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input: head = [1,2]
Output: [2,1]
Expected: [2,1]

Contribute a testcase

The screenshot shows a user's solution for reversing a singly linked list. The code is identical to the accepted version above. However, it fails the third test case, which expects the output [1] for the input [1].

Test Result:

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input: head = [1]
Output: [1]
Expected: [1]

Contribute a testcase

21)

C v Auto

```
33     temp->next=list1;
34     list1=list1->next;
35     temp=temp->next;
36 }
37
38 if(list1==NULL) temp->next=list2;
39 else temp->next=list1;
```

Saved

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

```
list1 = [1,2,4]
```

list2 = [1,3,4]

Output

```
[1,1,2,3,4,4]
```

Expected

```
[1,1,2,3,4,4]
```

Sannidhi Patil Access all features with our Premium subscription!

My Lists Notebook Progress Points

Try New Features Orders My Playgrounds Settings Appearance Sign Out

C v Auto

```
33     temp->next=list1;
34     list1=list1->next;
35     temp=temp->next;
36 }
37
38 if(list1==NULL) temp->next=list2;
39 else temp->next=list1;
```

Saved

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

```
list1 = []
```

list2 = []

Output

```
[]
```

Expected

```
[]
```

Sannidhi Patil Access all features with our Premium subscription!

My Lists Notebook Progress Points

Try New Features Orders My Playgrounds Settings Appearance Sign Out

C v Auto

```
33     temp->next=list1;
34     list1=list1->next;
35     temp=temp->next;
36 }
37
38 if(list1==NULL) temp->next=list2;
39 else temp->next=list1;
```

Saved

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

```
list1 = []
```

list2 = [0]

Output

```
[0]
```

Expected

```
[0]
```

Sannidhi Patil Access all features with our Premium subscription!

My Lists Notebook Progress Points

Try New Features Orders My Playgrounds Settings Appearance Sign Out

141)

Code Accepted

```
C v Auto
22 } 
23 
24 return true;
25 }
```

Saved

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

```
head = [3,2,0,-4]
```

pos = 1

Output

```
true
```

Expected

```
true
```

Code Accepted

```
C v Auto
22 } 
23 
24 return true;
25 }
```

Saved

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

```
head = [1,2]
```

pos = 0

Output

```
true
```

Expected

```
true
```

Code Accepted

```
C v Auto
22 } 
23 
24 return true;
25 }
```

Saved

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

```
head = [1]
```

pos = -1

Output

```
false
```

Expected

```
false
```

Ln 23.

142)

OUTPUT:

The image displays three separate screenshots of a programming interface, likely a C++ IDE or compiler, showing the execution results for different test cases. Each screenshot includes a code editor window, a user profile bar at the top right, and a 'Test Result' section below.

Screenshot 1 (Top Left): Shows the following code:

```
C v Auto
18     {
19         SLOW = SLOW->next;
20     }
21     return entry;
22 }
23
24 return NULL;
25 }
```

Test Result: Accepted (Runtime: 3 ms)

Input:
head = [3,2,0,-4]
pos = 1

Output:
tail connects to node index 1

Expected:
tail connects to node index 1

Screenshot 2 (Top Right): Shows the same code as above.

Test Result: Accepted (Runtime: 3 ms)

Input:
head = [1,2]
pos = 0

Output:
tail connects to node index 0

Expected:
tail connects to node index 0

Screenshot 3 (Bottom Left): Shows the same code as above.

Test Result: Accepted (Runtime: 3 ms)

Input:
head = [1]
pos = -1

Output:
no cycle

Expected:
no cycle

A small note at the bottom center says: "Contribute a testcase"