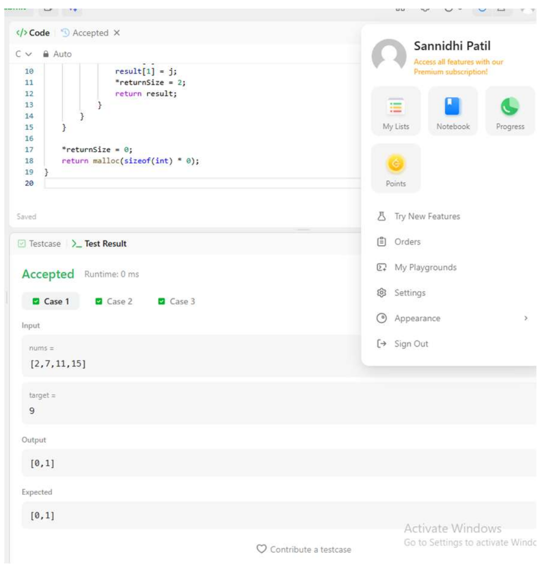
1)



```
            result[1] = j;
            *returnSize = 2;
            return result;
        }
    }
}

*returnSize = 0;
return malloc(sizeof(int) * 0);
}
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

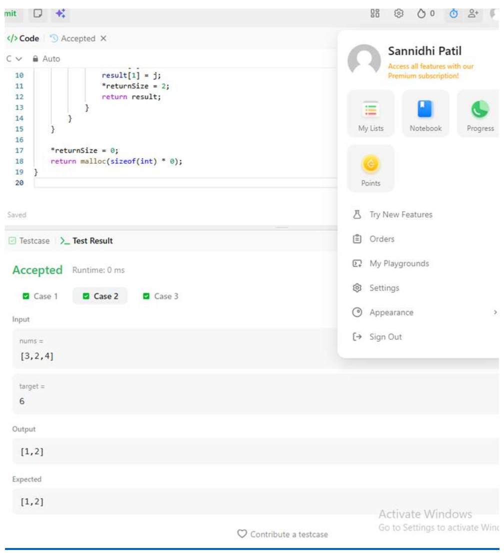
Input

nums =
[2,7,11,15]

target =
9

Output
[0,1]

Expected
[0,1]



```
            result[1] = j;
            *returnSize = 2;
            return result;
        }
    }
}

*returnSize = 0;
return malloc(sizeof(int) * 0);
}
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

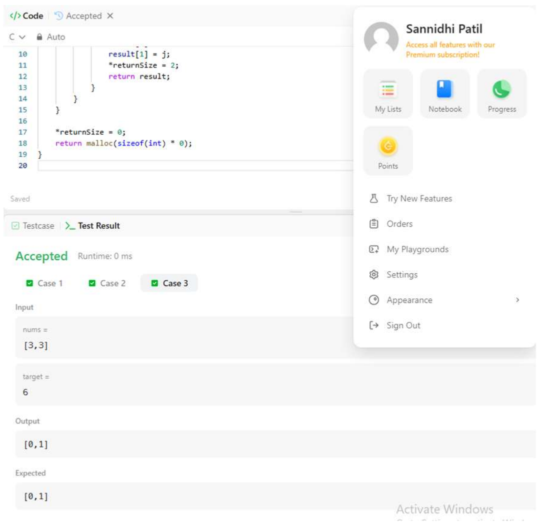
Input

nums =
[3,2,4]

target =
6

Output
[1,2]

Expected
[1,2]



```
            result[1] = j;
            *returnSize = 2;
            return result;
        }
    }
}

*returnSize = 0;
return malloc(sizeof(int) * 0);
}
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

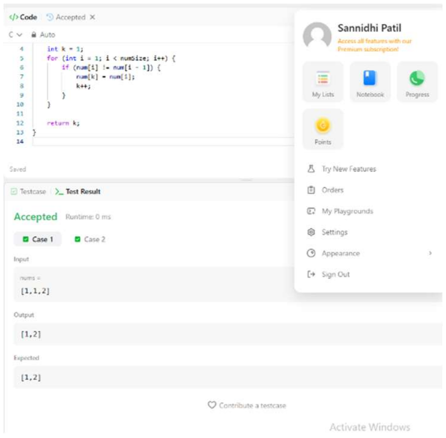
nums =
[3,3]

target =
6

Output
[0,1]

Expected
[0,1]

26)



```c
    int k = 1;
    for (int i = 1; i < numSize; i++) {
        if (num[i] != num[i - 1]) {
            num[k] = num[i];
            k++;
        }
    }

    return k;
}
```

Saved

Accepted Runtime: 0 ms

Input

nums =

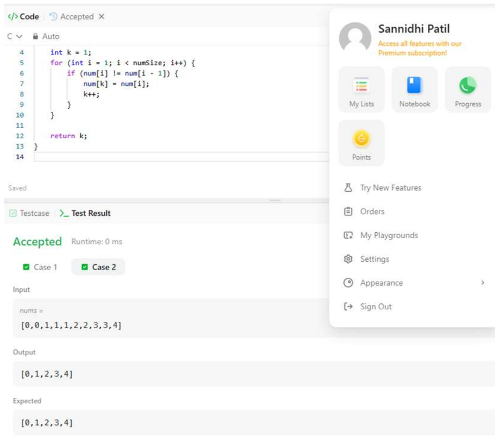
```
[1,1,2]
```

Output

```
[1,2]
```

Expected

```
[1,2]
```



```c
    int k = 1;
    for (int i = 1; i < numSize; i++) {
        if (num[i] != num[i - 1]) {
            num[k] = num[i];
            k++;
        }
    }

    return k;
}
```

Saved

Accepted Runtime: 0 ms

Input

nums =

```
[0,0,1,1,1,2,2,3,3,4]
```

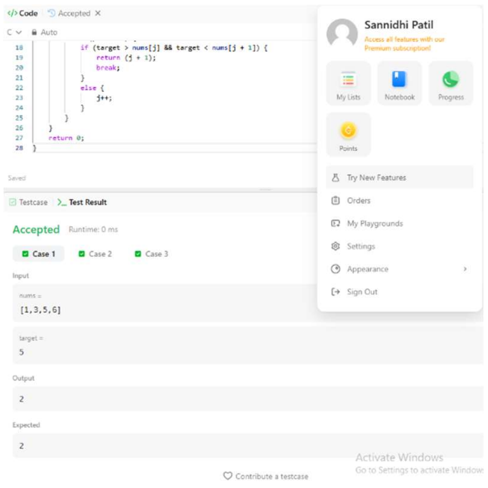
Output

```
[0,1,2,3,4]
```

Expected

```
[0,1,2,3,4]
```

35)



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

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums =
[1,3,5,6]

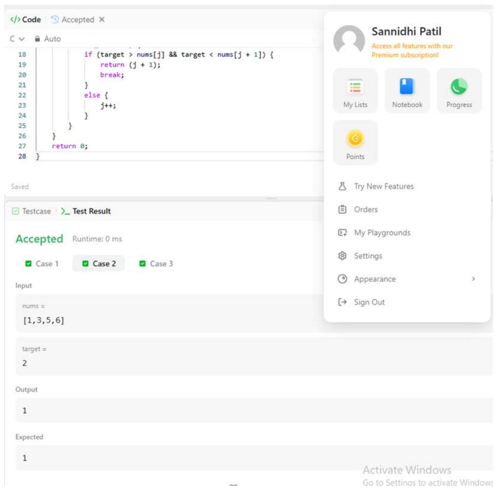
target =
5

Output

2

Expected

2



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

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums =
[1,3,5,6]

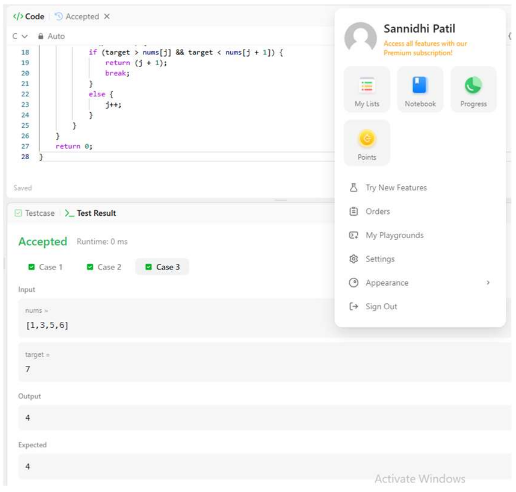
target =
2

Output

1

Expected

1



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

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

nums =
[1,3,5,6]

target =
7

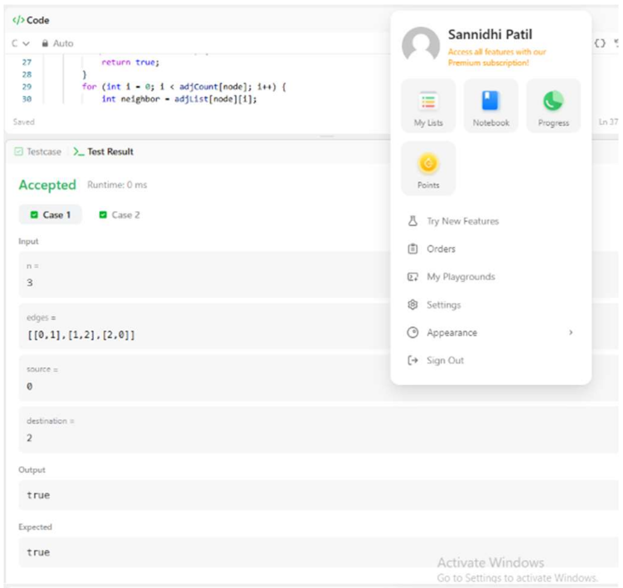
Output

4

Expected

4

1971)

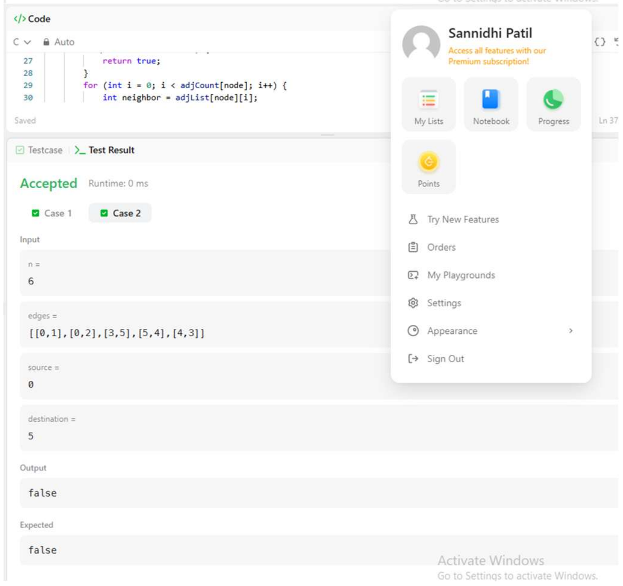


```
            return true;
        }
        for (int i = 0; i < adjCount[node]; i++) {
            int neighbor = adjList[node][i];
```

Accepted Runtime: 0 ms

Input

n =
```
3
```
edges =
```
[[0,1],[1,2],[2,0]]
```
source =
```
0
```
destination =
```
2
```
Output
```
true
```
Expected
```
true
```



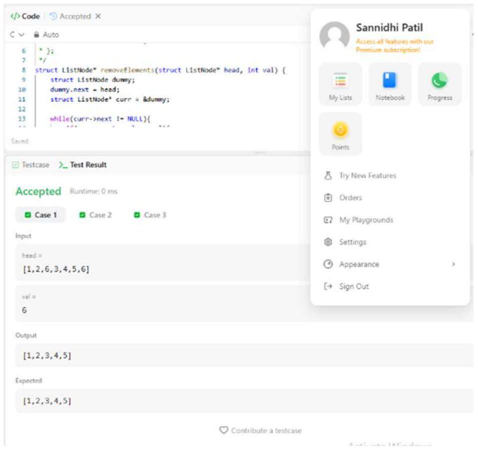
```
            return true;
        }
        for (int i = 0; i < adjCount[node]; i++) {
            int neighbor = adjList[node][i];
```

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
```

203)

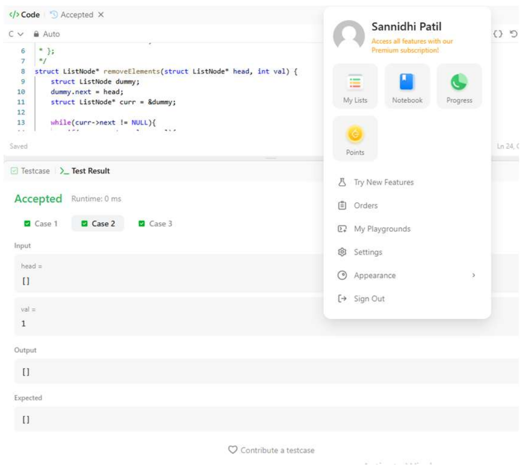


```
* };
*/
struct ListNode* removeElements(struct ListNode* head, int val) {
    struct ListNode dummy;
    dummy.next = head;
    struct ListNode* curr = &dummy;

    while(curr->next != NULL){
```

Accepted Runtime: 0 ms

Input

head =
```
[1,2,6,3,4,5,6]
```
val =
```
6
```
Output
```
[1,2,3,4,5]
```
Expected
```
[1,2,3,4,5]
```

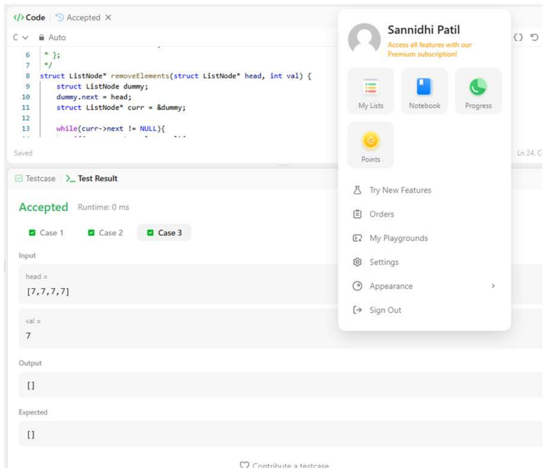


```
* };
*/
struct ListNode* removeElements(struct ListNode* head, int val) {
    struct ListNode dummy;
    dummy.next = head;
    struct ListNode* curr = &dummy;

    while(curr->next != NULL){
```

Accepted Runtime: 0 ms

Input

head =
```
[]
```
val =
```
1
```
Output
```
[]
```
Expected
```
[]
```



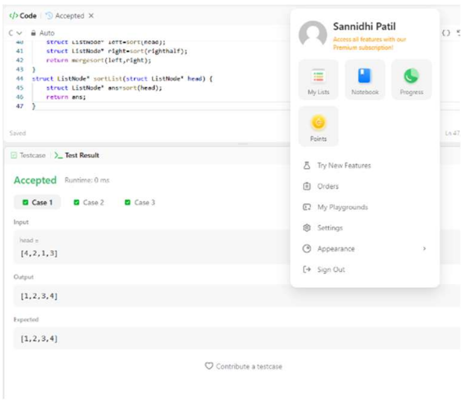
```
* };
*/
struct ListNode* removeElements(struct ListNode* head, int val) {
    struct ListNode dummy;
    dummy.next = head;
    struct ListNode* curr = &dummy;

    while(curr->next != NULL){
```

Accepted Runtime: 0 ms

Input

head =
```
[7,7,7,7]
```
val =
```
7
```
Output
```
[]
```
Expected
```
[]
```

148)



```
    struct ListNode* left=sort(head);
    struct ListNode* right=sort(righthalf);
    return mergesort(left,right);
}
struct ListNode* sortList(struct ListNode* head) {
    struct ListNode* ans=sort(head);
    return ans;
}
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

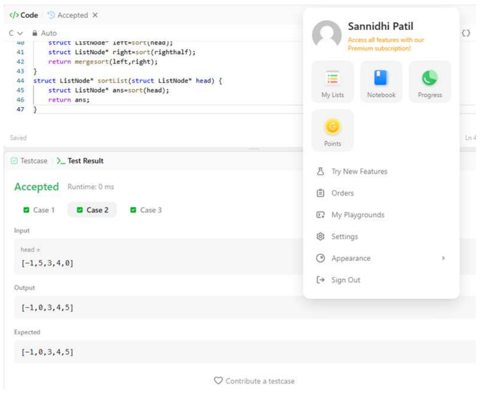
```
head =
[4,2,1,3]
```

Output

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

Expected

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



```
    struct ListNode* left=sort(head);
    struct ListNode* right=sort(righthalf);
    return mergesort(left,right);
}
struct ListNode* sortList(struct ListNode* head) {
    struct ListNode* ans=sort(head);
    return ans;
}
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

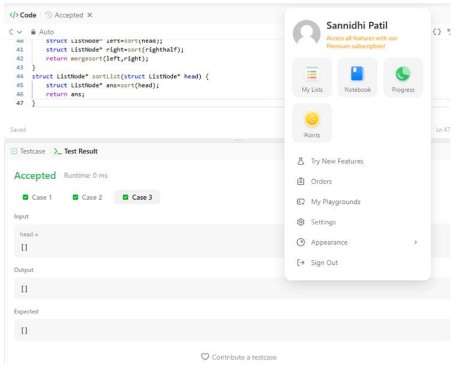
```
head =
[-1,5,3,4,0]
```

Output

```
[-1,0,3,4,5]
```

Expected

```
[-1,0,3,4,5]
```



```
    struct ListNode* left=sort(head);
    struct ListNode* right=sort(righthalf);
    return mergesort(left,right);
}
struct ListNode* sortList(struct ListNode* head) {
    struct ListNode* ans=sort(head);
    return ans;
}
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

```
head =
[]
```

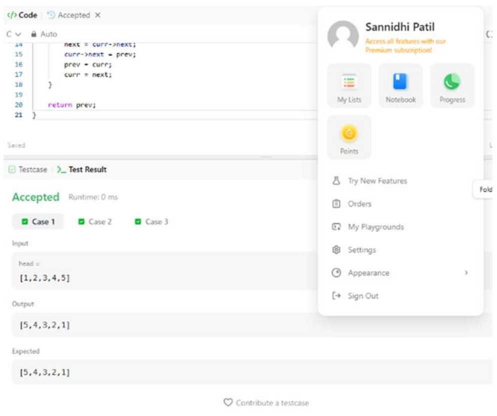
Output

```
[]
```

Expected

```
[]
```

206)



```
            next = curr->next;
            curr->next = prev;
            prev = curr;
            curr = next;
        }

        return prev;
}
```

Accepted Runtime: 0 ms

Input

head =

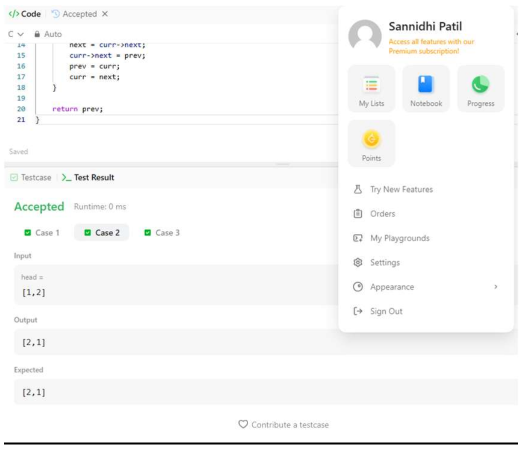
```
[1,2,3,4,5]
```

Output

```
[5,4,3,2,1]
```

Expected

```
[5,4,3,2,1]
```



```
            next = curr->next;
            curr->next = prev;
            prev = curr;
            curr = next;
        }

        return prev;
}
```

Accepted Runtime: 0 ms

Input

head =

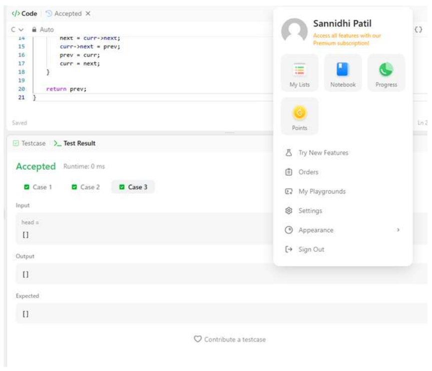
```
[1,2]
```

Output

```
[2,1]
```

Expected

```
[2,1]
```



```
            next = curr->next;
            curr->next = prev;
            prev = curr;
            curr = next;
        }

        return prev;
}
```

Accepted Runtime: 0 ms

Input

head =

```
[]
```

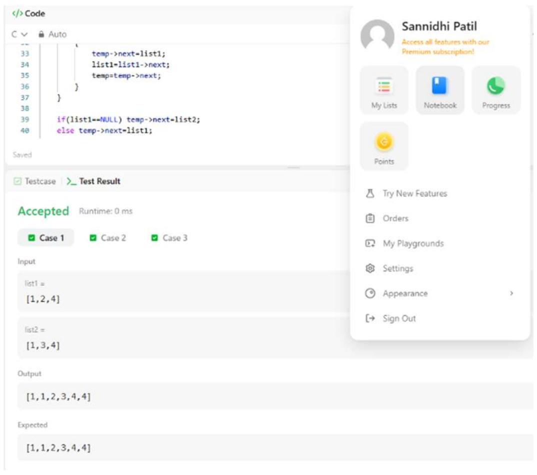
Output

```
[]
```

Expected

```
[]
```

21)

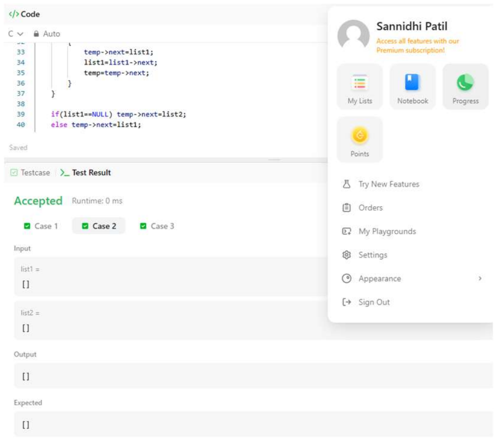


```
            temp->next=list1;
            list1=list1->next;
            temp=temp->next;
        }
    }

    if(list1==NULL) temp->next=list2;
    else temp->next=list1;
```

Accepted Runtime: 0 ms

Input

list1 =
```
[1,2,4]
```
list2 =
```
[1,3,4]
```
Output
```
[1,1,2,3,4,4]
```
Expected
```
[1,1,2,3,4,4]
```

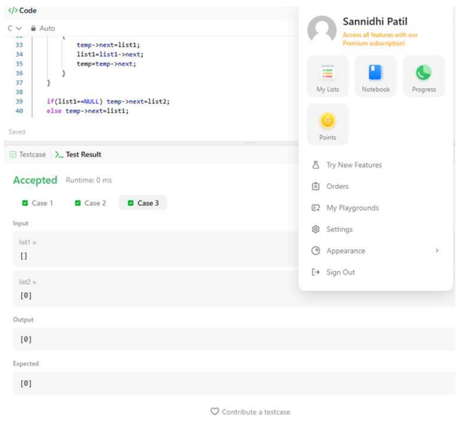


```
            temp->next=list1;
            list1=list1->next;
            temp=temp->next;
        }
    }

    if(list1==NULL) temp->next=list2;
    else temp->next=list1;
```

Accepted Runtime: 0 ms

Input

list1 =
```
[]
```
list2 =
```
[]
```
Output
```
[]
```
Expected
```
[]
```



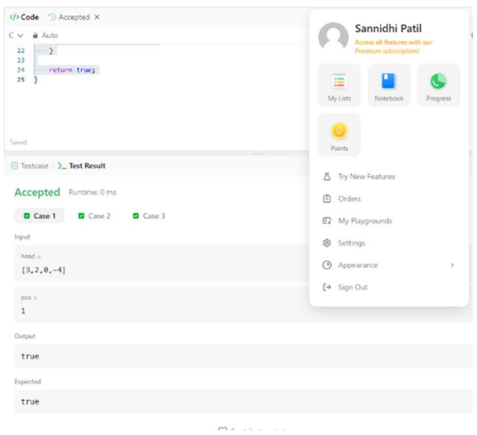
```
            temp->next=list1;
            list1=list1->next;
            temp=temp->next;
        }
    }

    if(list1==NULL) temp->next=list2;
    else temp->next=list1;
```

Accepted Runtime: 0 ms

Input

list1 =
```
[]
```
list2 =
```
[0]
```
Output
```
[0]
```
Expected
```
[0]
```

141)



```
    }

    return true;
}
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

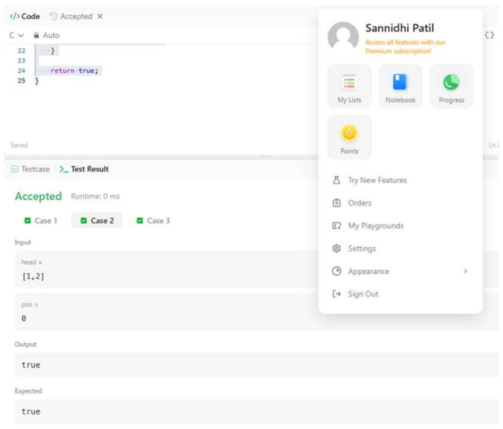
Input

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

pos =
```
1
```

Output
```
true
```

Expected
```
true
```



```
    }

    return true;
}
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

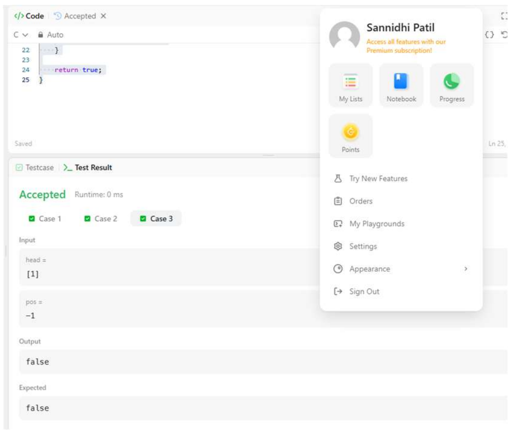
Input

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

pos =
```
0
```

Output
```
true
```

Expected
```
true
```



```
    }

    return true;
}
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

head =
```
[1]
```

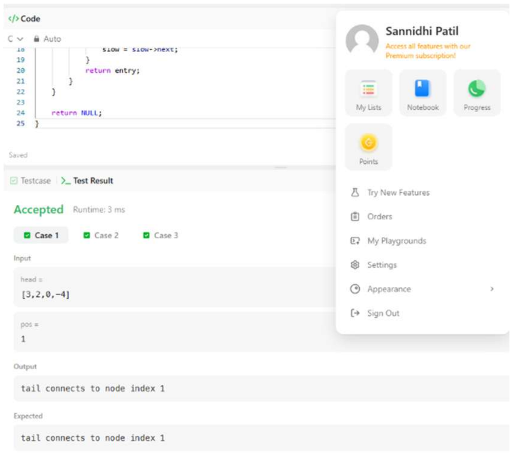
pos =
```
-1
```

Output
```
false
```

Expected
```
false
```

142)

OUTPUT:



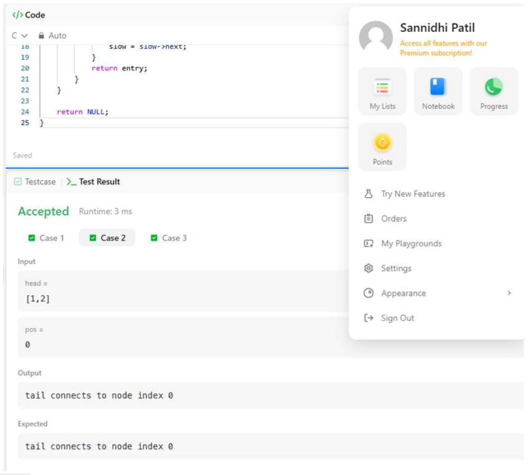
```
                slow = slow->next;
            }
            return entry;
        }
    }

    return NULL;
}
```

Accepted Runtime: 3 ms

Case 1 Case 2 Case 3

Input

head =
```
[3,2,0,-4]
```
pos =
```
1
```
Output
```
tail connects to node index 1
```
Expected
```
tail connects to node index 1
```



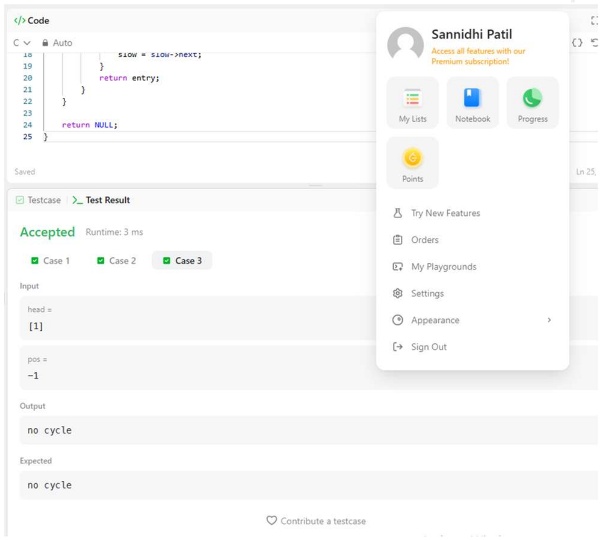
```
                slow = slow->next;
            }
            return entry;
        }
    }

    return NULL;
}
```

Accepted Runtime: 3 ms

Case 1 Case 2 Case 3

Input

head =
```
[1,2]
```
pos =
```
0
```
Output
```
tail connects to node index 0
```
Expected
```
tail connects to node index 0
```



```
                slow = slow->next;
            }
            return entry;
        }
    }

    return NULL;
}
```

Accepted Runtime: 3 ms

Case 1 Case 2 Case 3

Input

head =
```
[1]
```
pos =
```
-1
```
Output
```
no cycle
```
Expected
```
no cycle
```