LAB PRACTICALS (JAN-JUN, 2025)

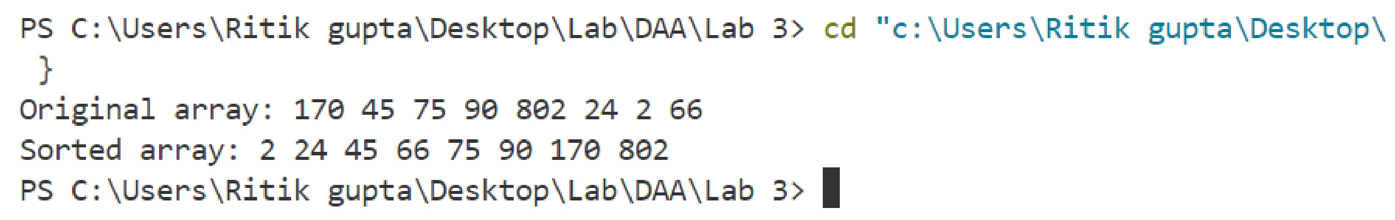
S. No.	WEEK No.	NAME OF PRACTICAL	Date	Signature

Lab – 03

Task – 01:

1.Radix sort:

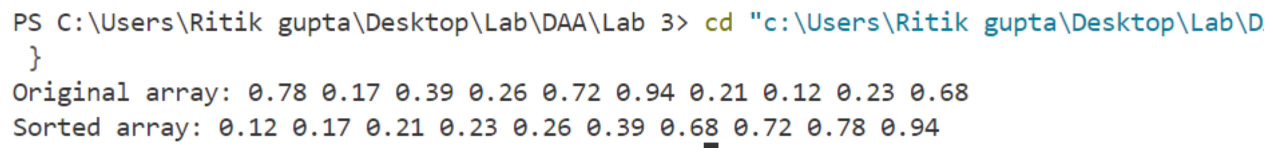
```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int maximum(const vector<int>& arr) {
    int maxi = 0;
    for(int i : arr){
        maxi = max(maxi,i);
    }
    return maxi;
}
void countingSort(vector<int>& arr, int exp) {
    int n = arr.size();
    vector<int> output(n);
    int count[10] = {0};

    for (int i = 0; i < n; i++)
        count[(arr[i] / exp) % 10]++;

    for (int i = 1; i < 10; i++)
        count[i] += count[i - 1];

    for (int i = n - 1; i >= 0; i--) {
        output[count[(arr[i] / exp) % 10] - 1] = arr[i];
        count[(arr[i] / exp) % 10]--;
    }
    for (int i = 0; i < n; i++)
        arr[i] = output[i];
}
void radixSort(vector<int>& arr) {
    int maxNum = maximum(arr);

    for (int exp = 1; maxNum / exp > 0; exp *= 10)
        countingSort(arr, exp);
}
void printArray(const vector<int>& arr) {
    for (int num : arr)
        cout << num << " ";
    cout << endl;
}
int main() {
    vector<int> arr = {170, 45, 75, 90, 802, 24, 2, 66};
    cout << "Original array: ";
    printArray(arr);

    radixSort(arr);
    cout << "Sorted array: ";
    printArray(arr);
    return 0;
}
```



```
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 3> cd "c:\Users\Ritik gupta\Desktop\
 }
Original array: 170 45 75 90 802 24 2 66
Sorted array: 2 24 45 66 75 90 170 802
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 3>
```

2.Bucket Sort:

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
void bucketSort(vector<float>& arr) {
    int n = arr.size();
    vector<vector<float>> buckets(n);
    for (float num : arr) {
        int index = n * num;
        buckets[index].push_back(num);
    }
    for (auto& bucket : buckets)
        sort(bucket.begin(), bucket.end());
    int idx = 0;
    for (auto& bucket : buckets)
        for (float num : bucket)
            arr[idx++] = num;
}
void printArray(const vector<float>& arr) {
    for (float num : arr)
        cout << num << " ";
    cout << endl;
}
int main() {
    vector<float> arr = {0.78, 0.17, 0.39, 0.26, 0.72, 0.94, 0.21, 0.12, 0.23, 0.68};
    cout << "Original array: ";
    printArray(arr);
    bucketSort(arr);
    cout << "Sorted array: ";
    printArray(arr);
    return 0;
}
```



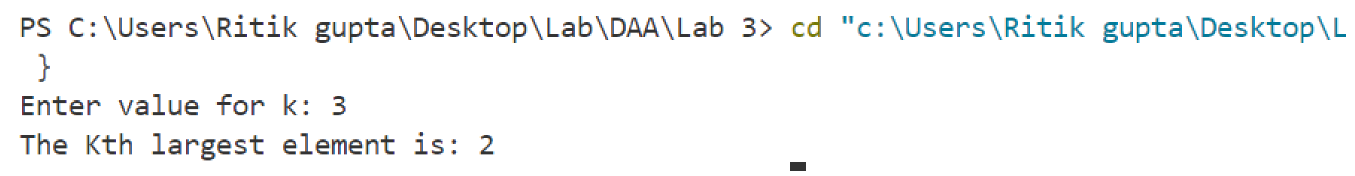
```
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 3> cd "c:\Users\Ritik gupta\Desktop\Lab\D
 }
Original array: 0.78 0.17 0.39 0.26 0.72 0.94 0.21 0.12 0.23 0.68
Sorted array: 0.12 0.17 0.21 0.23 0.26 0.39 0.68 0.72 0.78 0.94
```

Task-02

1.Kth Largest element in array:

```
#include<bits/stdc++.h>
```

```
using namespace std;
int findk(int arr[] , int size , int k){
    for(int i = 0; i<k ; i++){
        int maxi = INT_MIN;
        int max_idx = -1;
        for(int j = 0 ; j<size ; j++){
            if(arr[j] > maxi){
                maxi = arr[j];
                max_idx = j;}
            if(i == k-1){
                return maxi;
            }}
        arr[max_idx] = INT_MIN;}
    return -1;}
int main(){
    int arr[6] = {2,5,6,3,8,1};
    int n = 6;
    int k;
    cout<<"Enter value for k: ";
    cin>>k;

    int ans = findk(arr,n,k);

    cout<<"The Kth largest element is: "<<ans;}
```

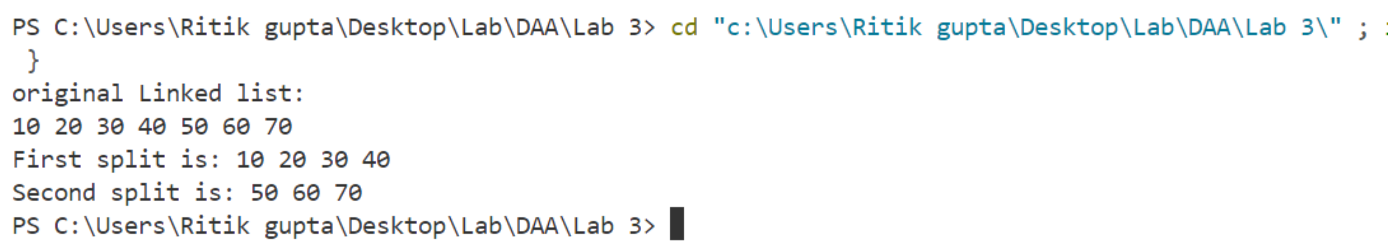


```
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 3> cd "c:\Users\Ritik gupta\Desktop\L
 }
Enter value for k: 3
The Kth largest element is: 2
```

2.Split linked list from middle:

```
#include<bits/stdc++.h>
using namespace std;
class Node{
    public:
    int data;
    Node* next;
    Node(int data){
        this->data = data;
        next  = NULL;
    }};
void insert(Node* &head,int data){
   Node* temp= new Node(data);
   if(head == NULL){
    head = temp;
    return;
   }
   Node* curr = head;
   while(curr->next!=NULL){
    curr = curr->next;
   }
   curr->next = temp;
}
```

```
void print(Node* head){
    while(head!=NULL){
        cout<<head->data<<" ";
        head = head->next;
    }
    cout<<endl;
}
void split(Node* head){
    Node* slow = head;
    Node* fast = head->next;
    while(fast!=NULL && fast->next!=NULL){
        slow  = slow ->next;
        fast = fast ->next->next;
    }
    Node* head2 = slow->next;
    slow->next = NULL;
    cout<<"First split is: ";
    print(head);
    cout<<"Second split is: ";
    print(head2);
}
int main(){
    Node* head = new Node(10);
    insert(head,20);
    insert(head,30);
    insert(head,40);
    insert(head,50);
    insert(head,60);
    insert(head,70);
    cout<<"original Linked list: "<<endl;
    print(head);
    split(head);
}
```

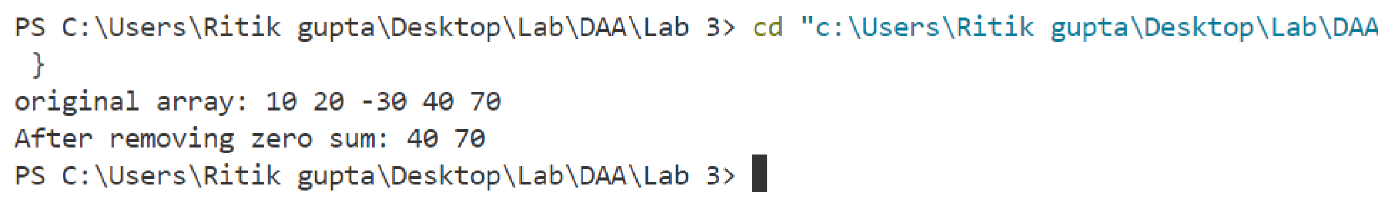


```
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 3> cd "c:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 3\" ;
 }
original Linked list:
10 20 30 40 50 60 70
First split is: 10 20 30 40
Second split is: 50 60 70
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 3>
```

3.Remove consecutive node with sum zero:

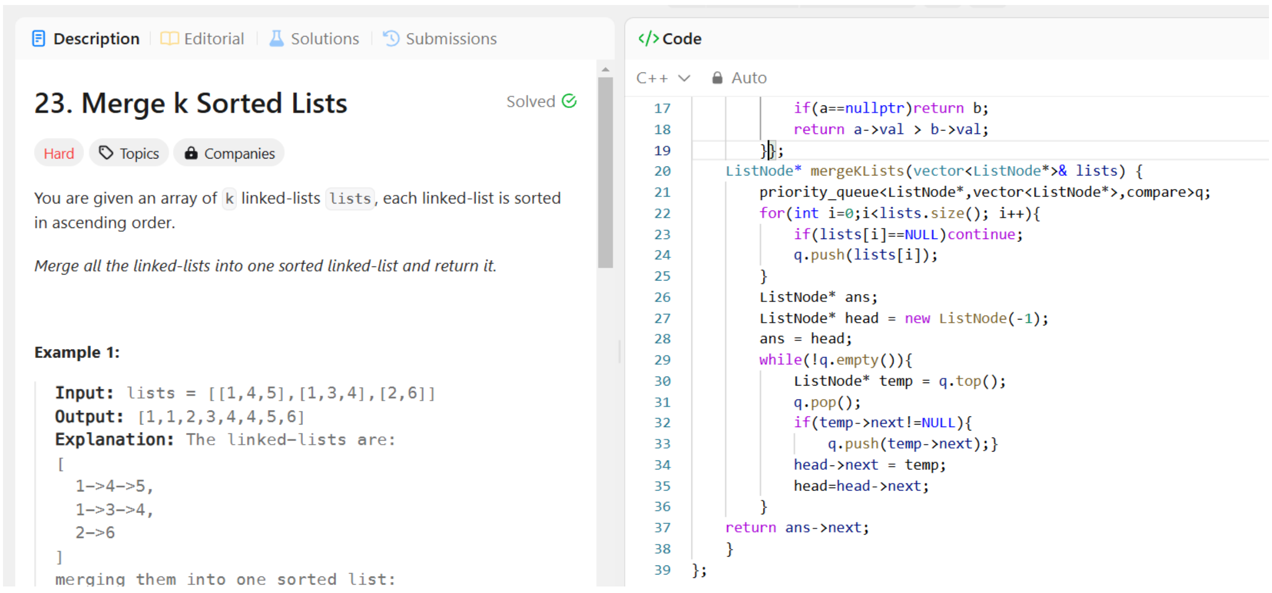
```
#include<bits/stdc++.h>
using namespace std;
class Node{
    public:
    int data;
    Node* next;
    Node(int data){
        this->data = data;
        next  = NULL;
    }
};
```

```cpp
void insert(Node* &head,int data){
   Node* temp= new Node(data);
   if(head == NULL){
    head = temp;
    return;
   }
   Node* curr = head;
   while(curr->next!=NULL){
    curr = curr->next;
   }
   curr->next = temp;
}
void print(Node* head){
    while(head!=NULL){
        cout<<head->data<<" ";
        head = head->next;
    }
    cout<<endl;
}
Node* remove_zero(Node* head){
    int sum = 0;
    Node* ans = head;
    while(head!=NULL){
        sum = sum + head->data;
        if(sum == 0){
            ans = head->next;
        }
        head = head->next;
    }
    return ans;
}
int main(){
    Node* head = new Node(10);
    insert(head,20);
    insert(head,-30);
    insert(head,40);
    insert(head,70);
    cout<<"original array: ";
    print(head);
    Node* ans = remove_zero(head);
    cout<<"After removing zero sum: ";
    print(ans);
}
```



```
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 3> cd "c:\Users\Ritik gupta\Desktop\Lab\DAA
 }
original array: 10 20 -30 40 70
After removing zero sum: 40 70
PS C:\Users\Ritik gupta\Desktop\Lab\DAA\Lab 3> 
```

1.Merge k sorted lists:



23. Merge k Sorted Lists

Solved

Hard · Topics · Companies

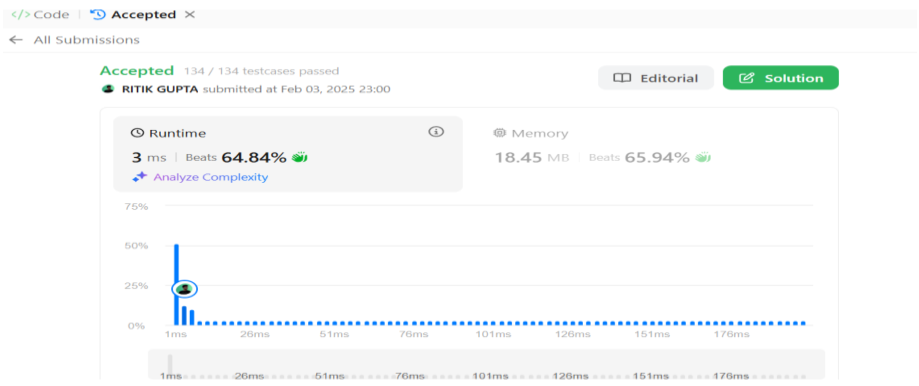
You are given an array of `k` linked-lists `lists`, each linked-list is sorted in ascending order.

Merge all the linked-lists into one sorted linked-list and return it.

Example 1:

```
Input: lists = [[1,4,5],[1,3,4],[2,6]]
Output: [1,1,2,3,4,4,5,6]
Explanation: The linked-lists are:
[
  1->4->5,
  1->3->4,
  2->6
]
merging them into one sorted list:
```

```cpp
            if(a==nullptr)return b;
            return a->val > b->val;
        }};
    ListNode* mergeKLists(vector<ListNode*>& lists) {
        priority_queue<ListNode*,vector<ListNode*>,compare>q;
        for(int i=0;i<lists.size(); i++){
            if(lists[i]==NULL)continue;
            q.push(lists[i]);
        }
        ListNode* ans;
        ListNode* head = new ListNode(-1);
        ans = head;
        while(!q.empty()){
            ListNode* temp = q.top();
            q.pop();
            if(temp->next!=NULL){
                q.push(temp->next);}
            head->next = temp;
            head=head->next;
        }
    return ans->next;
    }
};
```

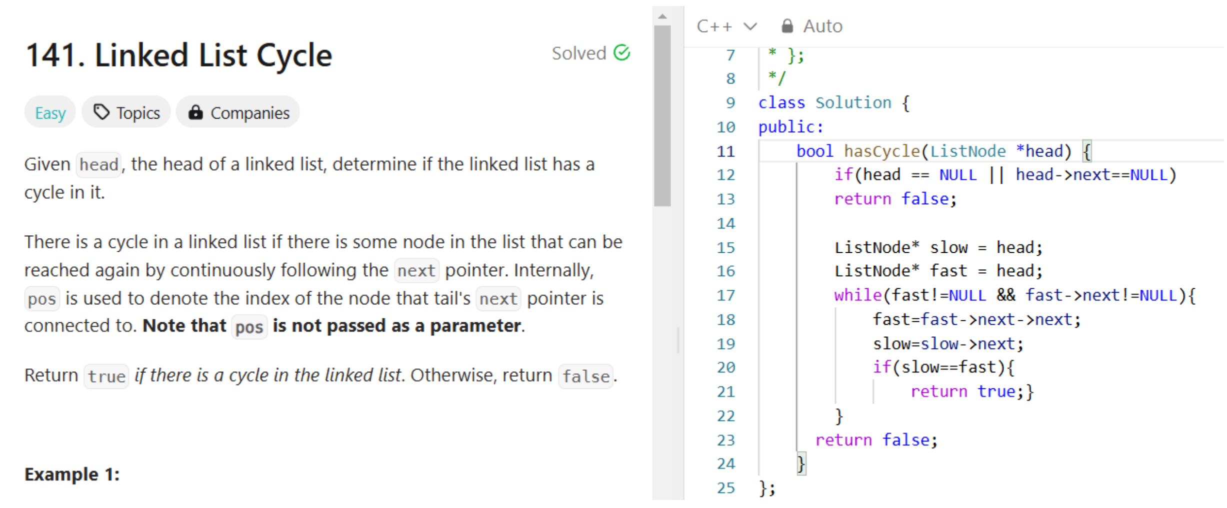


Accepted 134 / 134 testcases passed
RITIK GUPTA submitted at Feb 03, 2025 23:00

Runtime
3 ms | Beats 64.84%

Memory
18.45 MB | Beats 65.94%

2. Linked list cycle



141. Linked List Cycle

Solved

Easy · Topics · Companies

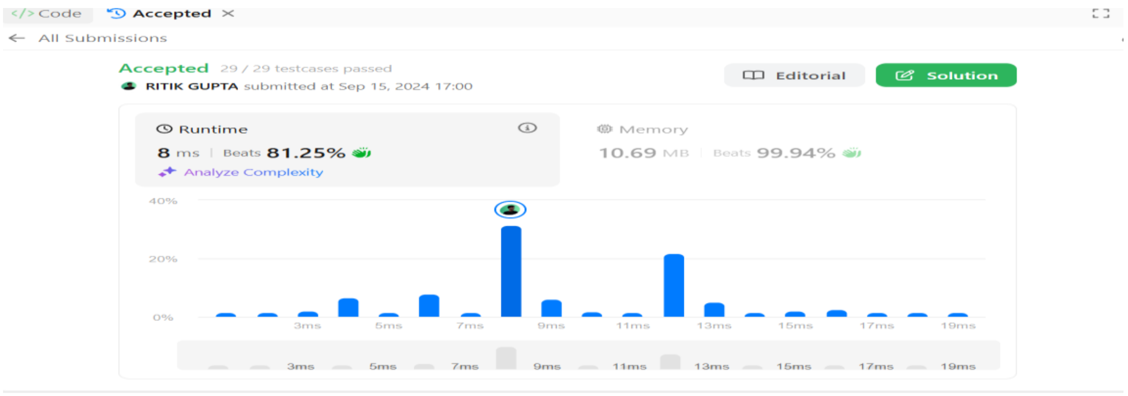
Given `head`, the head of a linked list, determine if the linked list has a cycle in it.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the `next` pointer. Internally, `pos` is used to denote the index of the node that tail's `next` pointer is connected to. **Note that `pos` is not passed as a parameter.**

Return `true` *if there is a cycle in the linked list*. Otherwise, return `false`.

Example 1:

```cpp
 * };
 */
class Solution {
public:
    bool hasCycle(ListNode *head) {
        if(head == NULL || head->next==NULL)
        return false;

        ListNode* slow = head;
        ListNode* fast = head;
        while(fast!=NULL && fast->next!=NULL){
            fast=fast->next->next;
            slow=slow->next;
            if(slow==fast){
                return true;}
        }
       return false;
    }
};
```



Accepted 29 / 29 testcases passed
RITIK GUPTA submitted at Sep 15, 2024 17:00

Runtime
8 ms | Beats 81.25%

Memory
10.69 MB | Beats 99.94%