|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LAB PRACTICALS (JAN-JUN, 2025)     |  |  |  |  |  | | --- | --- | --- | --- | --- | |  |  | NAME OF PRACTICAL | Date | Signature | | S.  No. | WEEK  No. |  |  |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  |  |  | |  |  |  |  |  | |

**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);

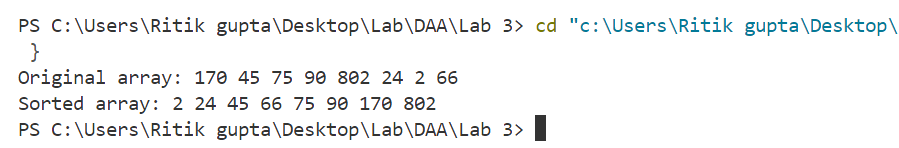
    radixSort(arr);

    cout << "Sorted array: ";

    printArray(arr);

    return 0;

}

****

**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);

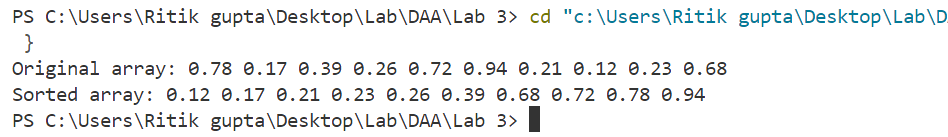
    bucketSort(arr);

    cout << "Sorted array: ";

    printArray(arr);

    return 0;

}

****

**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;

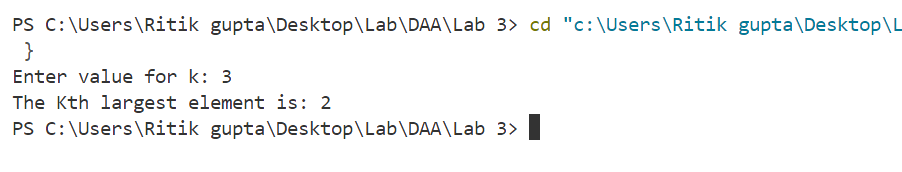
    int k;

    cout<<"Enter value for k: ";

    cin>>k;

    int ans = findk(arr,n,k);

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

****

**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);

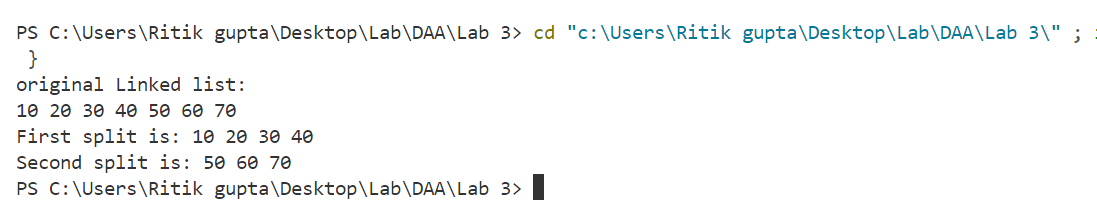
    insert(head,70);

    cout<<"original Linked list: "<<endl;

    print(head);

    split(head);

}

****

**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;

    }

};

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: ";

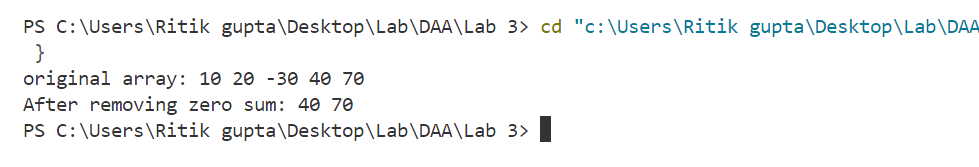
    print(head);

    Node\* ans = remove\_zero(head);

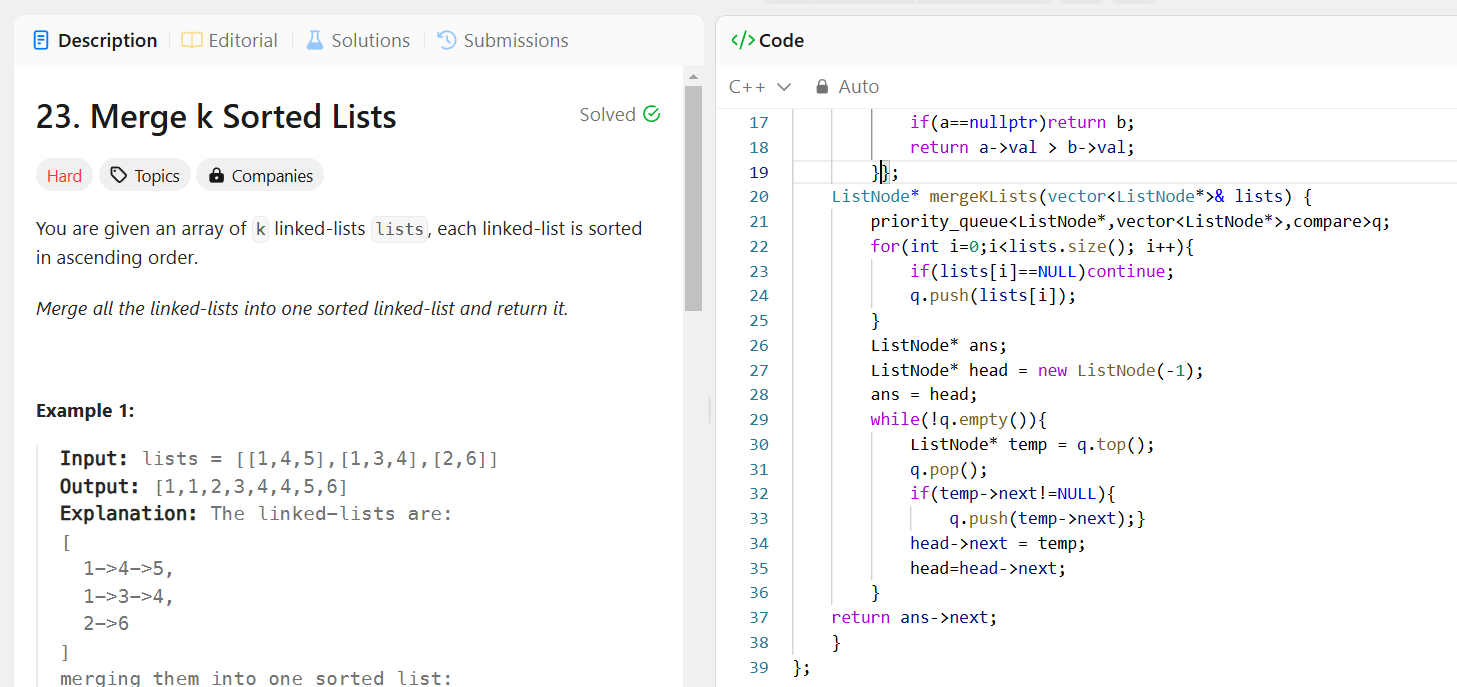
    cout<<"After removing zero sum: ";

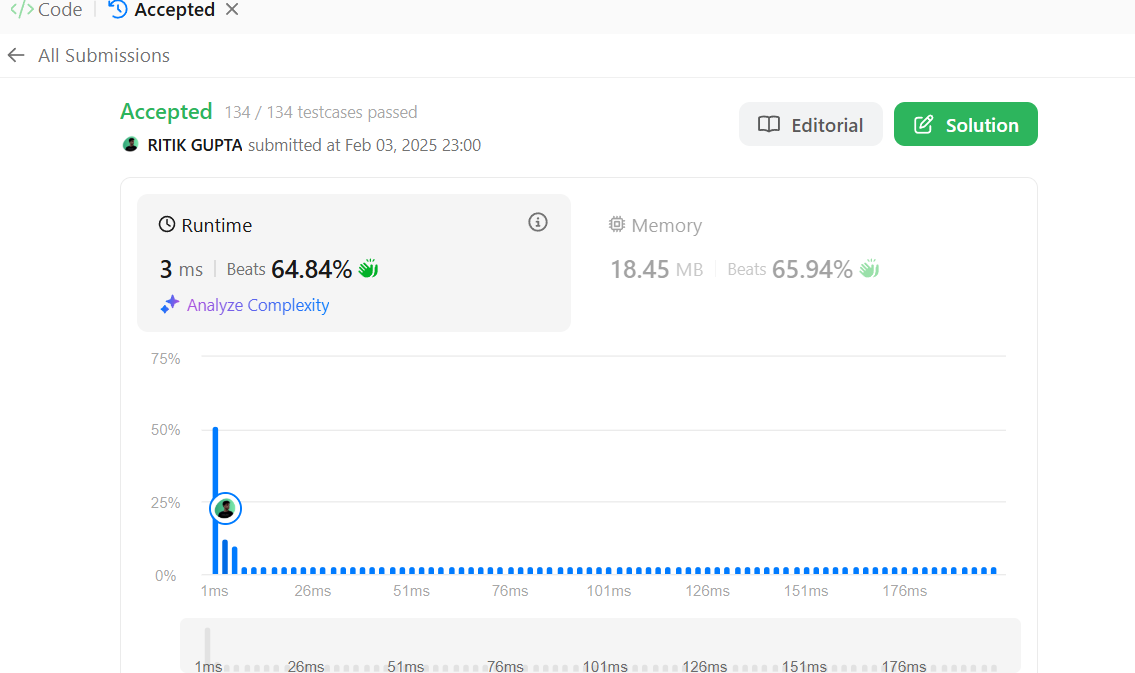
    print(ans);

}

****

**1.Merge k sorted lists:**

****



2. Linked list cycle