Question 3-

⁠Next Greater Element

Given a circular array, find the next greater number for every element.

Code-

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int nextGreater(int n) {

string digits = to\_string(n);

if (next\_permutation(digits.begin(), digits.end())) {

return stoi(digits);

} else {

return n; // If no greater permutation exists

}

}

int main() {

int n;

cin>>n;

int res = nextGreater(n);

cout<< res;

return 0;

}

Output -



Question 4 -

Add Two Numbers (Linked List)

You are given two non-empty linked lists representing two non-negative integers. Add the two numbers and return the sum as a linked list.

#include <bits/stdc++.h>

using namespace std;

struct ListNode {

int val;

ListNode\* next;

ListNode(int x) : val(x), next(nullptr) {}

};

ListNode\* addTwoNumbers(ListNode\* l1, ListNode\* l2) {

ListNode\* dummy = new ListNode(0);

ListNode\* curr = dummy;

int carry = 0;

while (l1 || l2 || carry) {

int sum = carry;

if (l1) sum += l1->val, l1 = l1->next;

if (l2) sum += l2->val, l2 = l2->next;

carry = sum / 10;

curr->next = new ListNode(sum % 10);

curr = curr->next;

}

return dummy->next;

}

void printList(ListNode\* head) {

while (head) {

cout << head->val;

if (head->next) cout << " -> ";

head = head->next;

}

cout << endl;

}

int main() {

ListNode\* l1 = new ListNode(5);

l1->next = new ListNode(4);

l1->next->next = new ListNode(3);

l1->next->next->next = new ListNode(3);

ListNode\* l2 = new ListNode(5);

l2->next = new ListNode(7);

l2->next->next = new ListNode(4);

l2->next->next->next = new ListNode(7);

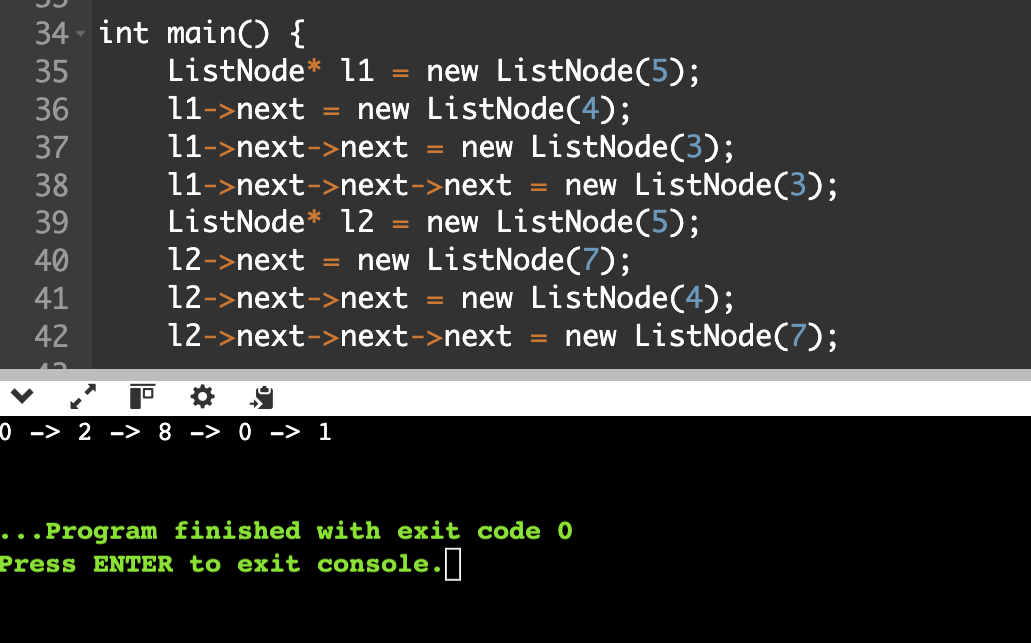
ListNode\* res = addTwoNumbers(l1, l2);

printList(res);

return 0;

}

Output -



Question 5 -

Reorder List

Reorder a linked list from L0,L1, ...,Ln-1 Ln to L0,Ln,L1,Ln-1-L2-Ln-2,.....

Code -

#include <bits/stdc++.h>

using namespace std;

struct ListNode {

int val;

ListNode\* next;

ListNode(int x) : val(x), next(nullptr) {}

};

void printList(ListNode\* head) {

while (head) {

cout << head->val;

if (head->next) cout << " -> ";

head = head->next;

}

cout << endl;

}

void reorderList(ListNode\* head) {

if (!head || !head->next) return;

//middle

ListNode \*slow = head, \*fast = head;

while (fast->next && fast->next->next) {

slow = slow->next;

fast = fast->next->next;

}

// Reverse

ListNode\* prev = nullptr;

ListNode\* curr = slow->next;

slow->next = nullptr;

while (curr) {

ListNode\* next = curr->next;

curr->next = prev;

prev = curr;

curr = next;

}

// Merge

ListNode\* first = head;

ListNode\* second = prev;

while (second) {

ListNode\* tmp1 = first->next;

ListNode\* tmp2 = second->next;

first->next = second;

second->next = tmp1;

first = tmp1;

second = tmp2;

}

}

int main() {

ListNode\* head = new ListNode(1);

head->next = new ListNode(2);

head->next->next = new ListNode(3);

head->next->next->next = new ListNode(4);

head->next->next->next->next = new ListNode(7);

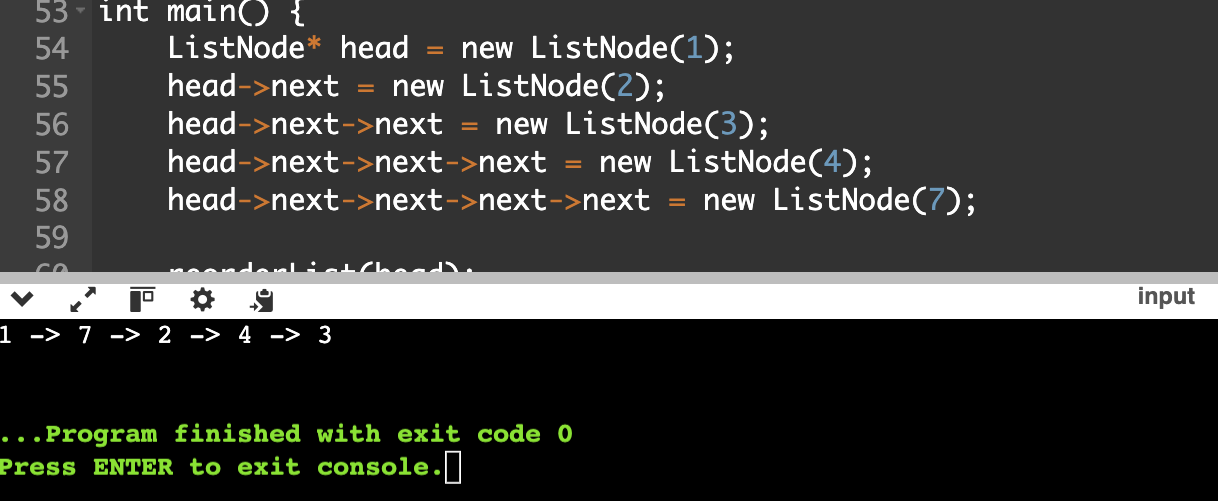
reorderList(head);

printList(head);

return 0;

}

Output -



Question 6 -

Given a linked list, remove the nth node from the end and return its head.

Code -

#include <bits/stdc++.h>

using namespace std;

struct ListNode {

int val;

ListNode\* next;

ListNode(int x) : val(x), next(nullptr) {}

};

void printList(ListNode\* head) {

while (head) {

cout << head->val;

if (head->next) cout << " -> ";

head = head->next;

}

cout << endl;

}

ListNode\* removeNthFromEnd(ListNode\* head, int n) {

ListNode\* dummy = new ListNode(0);

dummy->next = head;

ListNode\* first = dummy;

ListNode\* second = dummy;

for (int i = 0; i <= n; ++i) {

if (first) first = first->next;

}

// Move both pointers until first reaches end

while (first) {

first = first->next;

second = second->next;

}

// Delete nth last node

ListNode\* nodeToDelete = second->next;

second->next = second->next->next;

delete nodeToDelete;

ListNode\* newHead = dummy->next;

delete dummy;

return newHead;

}

int main() {

ListNode\* head = new ListNode(1);

head->next = new ListNode(2);

head->next->next = new ListNode(4);

head->next->next->next = new ListNode(4);

head->next->next->next->next = new ListNode(5);

head->next->next->next->next->next = new ListNode(3);

int n = 3;

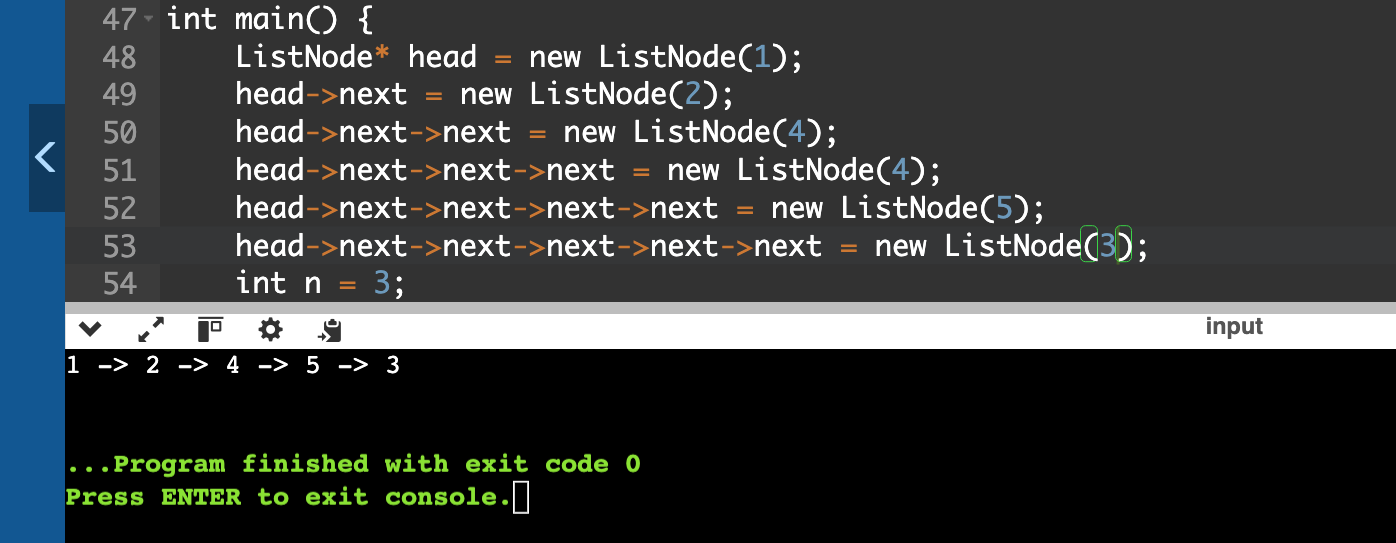
head = removeNthFromEnd(head, n);

printList(head);

return 0;

}

Output -



Question 8 -

Given an array of strings, group the anagrams together.

Code -

#include <bits/stdc++.h>

using namespace std;

string getCharCountKey(const string& str) {

int count[26] = {0};

for (char ch : str)

count[ch - 'a']++;

string key;

for (int i = 0; i < 26; i++) {

key += to\_string(count[i]) + "#";

}

return key;

}

// Group anagrams using character count keys

vector<vector<string>> groupAnagrams(vector<string>& strs) {

unordered\_map<string, vector<string>> mp;

for (const string& str : strs) {

string key = getCharCountKey(str);

mp[key].push\_back(str);

}

vector<vector<string>> result;

for (auto& [k, grp] : mp)

result.push\_back(grp);

return result;

}

int main() {

int n;

cin >> n;

vector<string> strs(n);

for (int i = 0; i < n; i++)

cin >> strs[i];

vector<vector<string>> result = groupAnagrams(strs);

for (auto& grp : result) {

for (auto& word : grp)

cout << word << " ";

cout << endl;

}

return 0;

}

Output -

