Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт №6**

***з дисципліни:*** «Мови та парадигми програмування»

***з розділу***: «Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур.»

***Виконала:***

студентка групи ШІ-11

Савчук Аня Григорівна

# **Тема роботи:**

Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур.

# **Мета роботи:**

Опрацювати теоретичний матеріал до тем засвоїти отримані знання при виконанні практичних та лабораторних робіт.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1:Черга, Стек.
* Тема №2: Списки
* Тема №3: Дерево

1. Індивідуальний план опрацювання теорії:

* Тема №1: Черга, Стек
  + Джерела Інформації
    - <https://youtu.be/ZYvYISxaNL0?si=alGKQ8qtvPVh-mCM>
    - <https://youtu.be/Yhw8NbjrSFA?si=kbi3AYlUXvLb1wB4>
  + Що опрацьовано:
    - Поняття черги та стеку.
  + Статус: Ознайомлений.
  + Початок опрацювання теми: 01.12.23
  + Звершення опрацювання теми: 08.12.23
* Тема №2: Списки
  + Джерела Інформації:
    - <https://www.youtube.com/watch?v=QLzu2-_QFoE>
    - <https://youtu.be/-25REjF_atI?si=CzHhgERDpJ6JA9vi>
  + Що опрацьовано:
    - Однозвязні списки. Двозвязні списки. Робота зі списками.
  + Статус: Ознайомлений.
  + Початок опрацювання теми: 01.12.23
  + Звершення опрацювання теми: 08.12.23
* Тема №3: Дерево
  + Джерела Інформації:
    - <https://youtu.be/qBFzNW0ALxQ?si=WUq1NoSbAXMD-XAc>
  + Що опрацьовано:
    - Бінарне дерево. Реалізація бінарного дерева.
  + Статус: Ознайомлений.
  + Початок опрацювання теми: 01.12.23
  + Звершення опрацювання теми: 08.12.23

# **Виконання роботи:**

## **1. Опрацювання завдання та вимог до програм та середовища:**

Завдання №1 **VNS Lab 10**

* Варіант завдання: **19**
* Деталі завдання:

Записи в лінійному списку містять ключове поле типу \*char (рядок символів). Сформувати двонаправлений список. Знищити К елементів із заданими номерами. Додати К елементів у початок списку.

Завдання №2 **Algotester Lab 5**

* Варіант завдання: **2**
* Деталі завдання:

В пустелі існує незвичайна печера, яка є двохвимірною. Її висота це N , ширина - M . Всередині печери є пустота, пісок та каміння. Пустота позначається буквою О , пісок S і каміння X ; Одного дня стався землетрус і весь пісок посипався вниз. Він падає на найнижчу клітинку з пустотою, але він не може пролетіти через каміння. Ваше завдання сказати як буде виглядати печера після землетрусу.

* Важливі деталі для врахування в імплементації програми

1≤N,M≤1000

Завдання №3 **Algotester Lab7\_8**

* Варіант завдання: **1**
* Деталі завдання

Ваше завдання - власноруч реалізувати структуру даних "Двозв’язний список". Ви отримаєте Q запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його аргументи. Вам будуть поступати запити такого типу:

Вставка: Ідентифікатор - i n s e r t Ви отримуєте ціле число i n d e x елемента, на місце якого робити вставку. Після цього в наступному рядку рядку написане число N - розмір списку, який треба вставити. У третьому рядку N цілих чисел - список, який треба вставити на позицію i n d e x .

Видалення: Ідентифікатор - e r a s e Ви отримуєте 2 цілих числа - i n d e x , індекс елемента, з якого почати видалення та n - кількість елементів, яку треба видалити.

Визначення розміру: Ідентифікатор - s i z e Ви не отримуєте аргументів. Ви виводите кількість елементів у списку.

Отримання значення i -го елементу Ідентифікатор - g e t Ви отримуєте ціле число - i n d e x , індекс елемента. Ви виводите значення елемента за індексом.

Модифікація значення i -го елементу Ідентифікатор - s e t Ви отримуєте 2 цілих числа - індекс елемента, який треба змінити, та його нове значення.

Вивід списку на екран Ідентифікатор - p r i n t Ви не отримуєте аргументів. Ви виводите усі елементи списку через пробіл. Реалізувати використовуючи перегрузку оператора <<

* Важливі деталі для врахування в імплементації програми:

0≤Q≤103

0≤lі≤103

Завдання №4 **Class Practice**

* Деталі завдання:

Реалізувати метод реверсу списку: Node\* reverse(Node \*head);

Реалізувати порівняння списків bool compare(Node \*h1, Node \*h2);

Реалізувати додавання великих чисел Node\* add(Node \*n1, Node \*n2);

Реалізувати віддзеркалення дерева TreeNode \*create\_mirror\_flip(TreeNode \*root);

Записати кожному батьківському вузлу суму підвузлів void tree\_sum(TreeNode \*root);

Завдання №5 **Self Practice**

* Деталі завдання:

Вам дана стрічка s. Ваше завдання зробити компресію стрічки, тобто якщо якась буква йде більше одного разу підряд у стрічці замінити її на букву + кількість входжень підряд.

## **2. Дизайн та планована оцінка часу виконання завдань:**

Програма №2 **Algotester Lab 5**

* Блок-схема

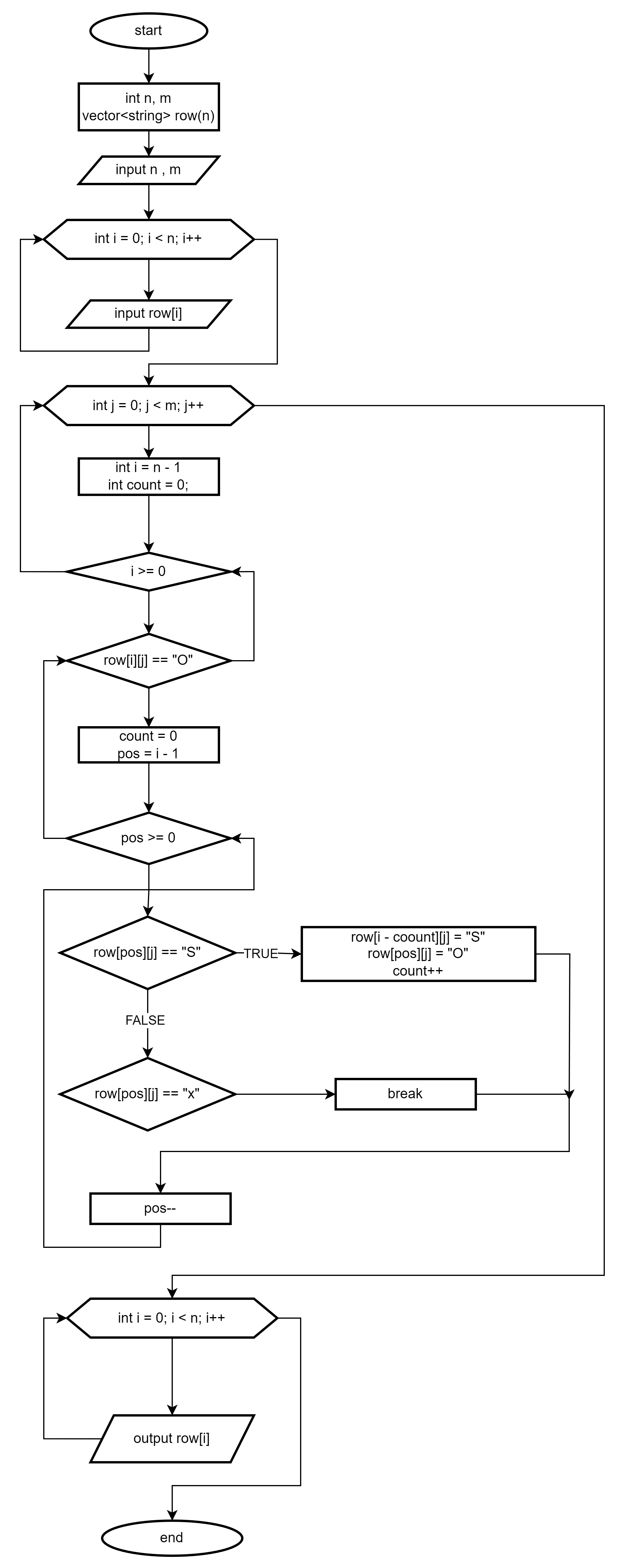


Image 1: Блок-схема до програми

* Планований час на реалізацію: 2 год

## **4. Код програм з посиланням на зовнішні ресурси:**

Завдання №1 **VNS Lab 10**

Посилання на файл програми у пул-запиті GitHub

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_savchuk_ania/ai_11/ania_savchuk/epic_6/vns_lab_10_task_1_variant_19_ania_savchuk.cpp>

#include <iostream>

#include <list>

using namespace std;

class DblLinkedList{

private:

    struct Node{

        int data;

        Node \*prev;

        Node \*next;

};

    Node \*head;

    Node \*tail;

    size\_t size;

public:

    void Show(){

        if(size == 0){

            cout << "List is empty" << endl;;

            return;

        }

        Node \*current = head;

        while (current != nullptr){

            cout << current->data << " ";

            current = current->next;

        }

        cout << endl;

}

    void PushBack(const int& value){

        if(size == 0){

            head = new Node{value, nullptr, nullptr};

            tail = head;

        }else{

            tail->next = new Node{ value, tail, nullptr};

            tail = tail->next;

        }

        size++;

    }

    void PushFront(int &value){

        if(size == 0){

            head = new Node{value, nullptr,nullptr};

            tail = head;

        }else{

            Node\* newHead = new Node{value, nullptr, head};

            head->prev = newHead;

            head = newHead;

        }

        size++;

}

    void Remove(int value) {

        Node\* current = head;

        while (current != nullptr) {

            if (current->data == value) {

                if (head == current) {

                    if (size == 1) {

                        delete head;

                        head = nullptr;

                        tail = nullptr;

                    } else {

                        Node\* tmp = head;

                        head = head->next;

                        head->prev = nullptr;

                        delete tmp;

                    }

                } else if (tail == current) {

                    Node\* tmp = tail;

                    tail = tail->prev;

                    tail->next = nullptr;

                    delete tmp;

                } else {

                    current->prev->next = current->next;

                    current->next->prev = current->prev;

                    delete current;

                }

                size--;

                return;

            }

            current = current->next;

        }

}

};

int main(){

    DblLinkedList list;

    int k = 3;

    cout << "List: " ;

    for (int value = 0; value < 10; value++){

        list.PushBack(value);

    }

list.Show();

    cout << "List after deletion: ";

    for (int i = 0; i < k; i++){

        list.Remove(i);

    }

    list.Show();

    cout << "List after adding elements: ";

    for (int i = 0; i < k ; i++){

        list.PushFront(i);

    }

list.Show();

}

Insert 1: Код до програми VNS Lab 10

Програма успішно працює відповідно до всіх вимог поставлених у задачі

Завдання №2 **Algotester Lab 5**

Посилання на файл програми у пул-запиті GitHub

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_savchuk_ania/ai_11/ania_savchuk/epic_6/algotester_lab_5_variant_2_ania_savchuk.cpp>

#include <iostream>

#include <vector>

using namespace std;

int main() {

    int n, m;

    cin >> n >> m;

    vector<string> row(n);

    for (int i = 0; i < n; i++) {

        cin >> row[i];

    }

    for (int j = 0; j < m; j++) {

        int i = n - 1;

        int count = 0;

        while (i >= 0) {

            if (row[i][j] == 'O') {

                count = 0;

                int pos = i - 1;

                while (pos >= 0) {

                    if (row[pos][j] == 'S') {

                        row[i-count][j] = 'S';

                        row[pos][j] = 'O';

                        count++;

                    } else if(row[pos][j] == 'X') {

                        break;

                    }

                    pos--;

                }

            }

            i--;

        }

    }

    for (int i = 0; i < n; i++) {

        cout << row[i] << endl;

    }

    return 0;

}

Insert 2: Код до програми Algotester Lab 5

Програма успішно працює відповідно до всіх вимог поставлених у задачі та пройшла перевірку на алготестері.

Завдання №3 **Algotester Lab7\_8**

Посилання на файл програми у пул-запиті GitHub

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_savchuk_ania/ai_11/ania_savchuk/epic_6/algotester_lab_7_8_variant_1_ania_savchuk.cpp>

#include <iostream>

using namespace std;

class DoublyLinkedList {

    private:

        struct Node {

            int data;

            Node\* next;

            Node\* prev;

        }

        Node\* head;

        Node\* tail;

        int size;

    public:

        DoublyLinkedList() {

            head = nullptr;

            tail = nullptr;

            size = 0;

        }

        void insert(int index, int N, int arr[]) {

            if (index > size) {

                return;

            }

            for (int i = N - 1; i >= 0; i--) {

                Node\* tmp = new Node();

                tmp->data = arr[i];

                tmp->prev = nullptr;

                tmp->next = nullptr;

                if (index == 0) {

                    tmp->next = head;

                    if (head != nullptr) {

                        head->prev = tmp;

                    }

                    head = tmp;

                    if (tail == nullptr) {

                        tail = tmp;

                    }

                } else if (index == size) {

                    tmp->prev = tail;

                    if (tail != nullptr) {

                        tail->next = tmp;

                    }

                    tail = tmp;

                    if (head == nullptr) {

                        head = tmp;

                    }

                } else {

                    Node\* current = head;

                    for (int j = 0; j < index - 1; j++) {

                        current = current->next;

                    }

                    tmp->next = current->next;

                    tmp->prev = current;

                    if (current->next != nullptr) {

                        current->next->prev = tmp;

                    }

                    current->next = tmp;

                }

                size++;

            }

        }

        void erase(int index, int n) {

            if (index >= size) {

                return;

            }

            Node\* current = head;

            while (index--) {

                current = current->next;

            }

            while (n--) {

                if (current->prev == nullptr) {

                    head = current->next;

                } else {

                    current->prev->next = current->next;

                }

                if (current->next == nullptr) {

                    tail = current->prev;

                } else {

                    current->next->prev = current->prev;

                }

                Node\* tmp = current;

                current = current->next;

                delete tmp;

                size --;

            }

        }

        int get(int index) {

            if (index >= size) {

                return -1;

            }

            Node\* current = head;

            while (index--) {

                current = current->next;

            }

            return current->data;

        }

        void set(int index, int val) {

            if (index >= size) {

                return;

            }

            Node\* current = head;

            while (index--) {

                current = current->next;

            }

            current->data = val;

        }

        size\_t Size() {

            return size;

        }

        friend ostream& operator<<(ostream& os, DoublyLinkedList& list) {

            Node\* current = list.head;

            while (current != nullptr) {

                os << current->data << " ";

                current = current->next;

            }

            return os;

        }

};

int main() {

    int Q;

    cin >> Q;

    DoublyLinkedList list;

    while (Q--) {

        string query;

        cin >> query;

        if (query == "insert") {

            int index, N;

            cin >> index >> N;

            int arr[N];

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

                cin >> arr[i];

            }

            list.insert(index, N, arr);

        } else if (query == "erase") {

            int index, n;

            cin >> index >> n;

            list.erase(index, n);

        } else if (query == "get") {

            int index;

            cin >> index;

            cout << list.get(index) << endl;

        } else if (query == "set") {

            int index, val;

            cin >> index >> val;

            list.set(index, val);

        } else if (query == "size") {

            cout << list.Size() << endl;

        } else if (query == "print") {

            cout << list << endl;

        }

    }

    return 0;

}

Insert 3: Код до програми Algotester Lab7\_8

Програма успішно працює відповідно до всіх вимог поставлених у задачі та пройшла перевірку на алготестері.

Завдання №4 **Class Practice**

Посилання на файл програми у пул-запиті GitHub

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_savchuk_ania/ai_11/ania_savchuk/epic_6/practice_work_task_1_ania_savchuk.cpp>

#include <iostream>

using  namespace std;

struct Node{

    int data;

Node \*next;

    Node(int val, Node \*node = nullptr ) {

        data = val;

        next = node;

    }

};

void pushFront(Node \*&head, int value){

    if(head == nullptr){

        head = new Node{value, nullptr};

    }else{

        Node \*newNode = new Node{value, head};

        head = newNode;

    }

}

void Show(Node \*head){

    if(head == nullptr){

        cout << "List is empty" << endl;;

        return;

}

    while (head != nullptr){

        cout << head->data << " ";

        head = head->next;

    }

    cout << endl;

}

Node \*reverse(Node \*head){

    Node \*current = head;

    Node \*prev = nullptr;

Node \*next = nullptr;

    while (current != nullptr){

        next = current->next;

        current->next = prev;

        prev =current;

        current = next;

    }

    return prev;

}

bool compare(Node \*h1, Node \*h2){

    while (h1 != nullptr && h2 != nullptr) {

        if (h1->data != h2->data) {

            // Знайдено невідповідність даних

            return false;

        }

        h1 = h1->next;

        h2 = h2->next;

}

    // Якщо один список закінчується раніше іншого, повертаємо false

    if (h1 != nullptr || h2 != nullptr) {

        return false;

}

    return true; // Списки однакові

}

Node\* add(Node \*num1, Node \*num2){

    Node \*result = nullptr;

    Node \*current = nullptr;

int carry = 0;

    while (num1 || num2 || carry){

        int val1 = (num1) ? num1->data : 0;

        int val2 = (num2) ? num2->data : 0;

        int sum = val1 + val2 + carry;

        int s = sum % 10;

        carry = sum / 10;

        if(result == nullptr){

            result = new Node(s);

            current = result;

        }else{

            current->next = new Node(s);

            current = current->next;

        }

        // Переходимо до наступних вузлів

        if (num1 != nullptr) num1 = num1->next;

        if (num2 != nullptr) num2 = num2->next;

}

    return result;

}

int main(){

    Node \*list = nullptr;

    pushFront(list, 7);

    pushFront(list, 1);

    pushFront(list, 10);

    pushFront(list, 3);

    pushFront(list, 5);

pushFront(list, 2);

    cout << "List: ";

Show(list);

list = reverse(list);

    cout << "Reversed list: ";

Show(list);

    Node \*list1 = nullptr;

    pushFront(list1, 5);

    pushFront(list1, 3);

    pushFront(list1, 10);

pushFront(list1, 9);

    cout << "List 1: ";

Show(list1);

    Node \*list2 = nullptr;

    pushFront(list2, 5);

    pushFront(list2, 0);

    pushFront(list2, 1);

pushFront(list2, 9);

    cout << "List 2: ";

Show(list2);

    bool result1 = compare(list1, list2);

    if(result1){

        cout << "List equal!" << endl;

    }else{

        cout << "List not equal!" << endl;

    }

    Node \*list3 = nullptr;

    pushFront(list3, 1);

    pushFront(list3, 2);

pushFront(list3, 3);

    cout << "List 3: ";

Show(list3);

    Node \*list4 = nullptr;

    pushFront(list4, 9);

    pushFront(list4, 8);

pushFront(list4, 7);

    cout << "List 4: ";

Show(list4);

    Node \*result2 = add(list3, list4);

result2 = reverse(result2);

    cout << "Sum: ";

Show(result2);

    return 0;

}

Insert 4: Код до програми Class Practice

Програма успішно працює відповідно до всіх вимог поставлених у задачі

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_savchuk_ania/ai_11/ania_savchuk/epic_6/practice_work_task_2_ania_savchuk.cpp>

#include <iostream>

#include <queue>

using  namespace std;

struct TreeNode{

    int data;

    TreeNode \*right;

TreeNode \*left;

    TreeNode(int val){

        data = val;

        left = nullptr;

        right = nullptr;

    }

};

TreeNode \*insert(TreeNode \*root, int data){

    if( root == nullptr){

        return new TreeNode(data);

}

    if( data > root->data){

        root->right = insert(root->right, data);

    }else if(data < root->data){

        root->left = insert(root->left, data);

}

    return root;

}

void show(TreeNode \*root){

    if(root == nullptr){

        return;

}

    show(root->left);

    cout << root->data << " ";

    show(root->right);

}

TreeNode \*create\_mirror\_flip(TreeNode \*root){

    if(root == nullptr){

        return nullptr;

    }

    // Створюємо новий вузол для нового дерева

    TreeNode\* mir\_root = new TreeNode(root->data);

    // Рекурсивно викликаємо функцію для лівого піддерева та присвоюємо його новому правому піддереву

mir\_root->left = create\_mirror\_flip(root->right);

    // Рекурсивно викликаємо функцію для правого піддерева та присвоюємо його новому лівому піддереву

mir\_root->right = create\_mirror\_flip(root->left);

    return mir\_root;

}

void tree\_sum(TreeNode \*root){

    if(root == nullptr){

        return;

}

    queue<TreeNode\*> node\_q;

node\_q.push(root);

    while (!node\_q.empty()){

        TreeNode \*current = node\_q.front();

        node\_q.pop();

        if (current->left != nullptr){

            node\_q.push(current->left);

            current->data += current->left->data;

        }

        if (current->right != nullptr){

            node\_q.push(current->right);

            current->data += current->right->data;

        }

    }

}

int main(){

    TreeNode \*root = nullptr;

    root = insert(root, 5);

    insert(root, 3);

    insert(root, 7);

    insert(root, 2);

    insert(root, 4);

    insert(root, 6);

insert(root, 8);

    cout << "TreeNode: ";

    show(root);

cout << endl;

    TreeNode \*mir\_tree = create\_mirror\_flip(root);

    cout << "New TreeNode: ";

    show(mir\_tree);

cout << endl;

tree\_sum(root);

    cout << "TreeNode after sum: ";

show(root);

    return 0;

}

Insert 5: Код до програми Class Practice

Програма успішно працює відповідно до всіх вимог поставлених у задачі

Завдання №5 **Self Practice**

Посилання на файл програми у пул-запиті GitHub

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_savchuk_ania/ai_11/ania_savchuk/epic_6/self_practice_work_algotester_task_1_ania_savchuk.cpp>

#include <iostream>

#include <string>

using namespace std;

int main(){

    string S;

    cin >> S;

int counter = 1;

    for (int i = 0; i < S.size(); i++){

        if (S[i] == S[i+1]){

            counter++;

        }else if(counter > 1){

            cout << S[i] << counter;

            counter = 1;

        }else {

            cout << S[i];

        }

    }

    return 0;

}

Insert 6: Код до програми Self Practice

Програма успішно працює відповідно до всіх вимог поставлених у задачі та пройшла перевірку на алготестері.

Завдання №

## **5. Результати виконання завдань, тестування та фактично затрачений час:**

Завдання №1 **VNS Lab 10**

INPUT/OUTPUT

List: 0 1 2 3 4 5 6 7 8 9

List after deletion: 3 4 5 6 7 8 9

List after adding elements: 2 1 0 3 4 5 6 7 8 9

Insert 7: Результат виконання програми VNS Lab 10

Час затрачений на виконання завдання

Завдання №2 **Algotester Lab 5**

INPUT/OUTPUT

5 5

SSOSS

OOOOO

SOOXX

OOOOS

OOSOO

OOOOO

OOOSS

OOOXX

SOOOO

SSSOS

Insert 8: Результат виконання програми Algotester Lab 5

Час затрачений на виконання завдання: 2 год



Image 2: Задачу зараховано на алготестері

Завдання №3 **Algotester Lab 7\_8**

INPUT/OUTPUT

9

insert

0

5

1 2 3 4 5

insert

2

3

7 7 7

print

1 2 7 7 7 3 4 5

erase

1 2

print

1 7 7 3 4 5

size

6

get

3

3

set

3 13

print

1 7 7 13 4 5

Insert 9: Результат виконання програми Algotester Lab 7\_8

Час затрачений на виконання завдання: 5 год

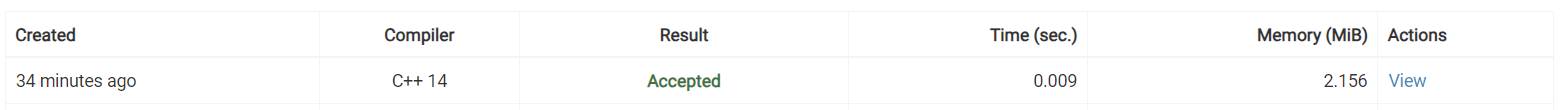


Image 3: Задачу зараховано на алготестері

Завдання №4 **Class Practice**

INPUT/OUTPUT

List: 2 5 3 10 1 7

Reversed list: 7 1 10 3 5 2

List 1: 9 10 3 5

List 2: 9 1 0 5

List not equal!

List 3: 3 2 1

List 4: 7 8 9

Sum: 1 1 1 0

Insert 10: Результат виконання програми Class Practice

Час затрачений на виконання завдання: 3 год

INPUT/OUTPUT

TreeNode: 2 3 4 5 6 7 8

New TreeNode: 8 7 6 5 4 3 2

TreeNode after sum: 2 9 4 15 6 21 8

Insert 11: Результат виконання програми Class Practice

Час затрачений на виконання завдання: 2 год

Завдання №5 **Self Practice**

INPUT/OUTPUT

AAAABBBCQQQQ

A4B3CQ4

Insert 12: Результат виконання програми Self Practice

Час затрачений на виконання завдання: 30 хв



Image 4: Задачу зараховано на алготестері

## **6. Кооперація з командою:**

* Скрін з 1-ї зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло

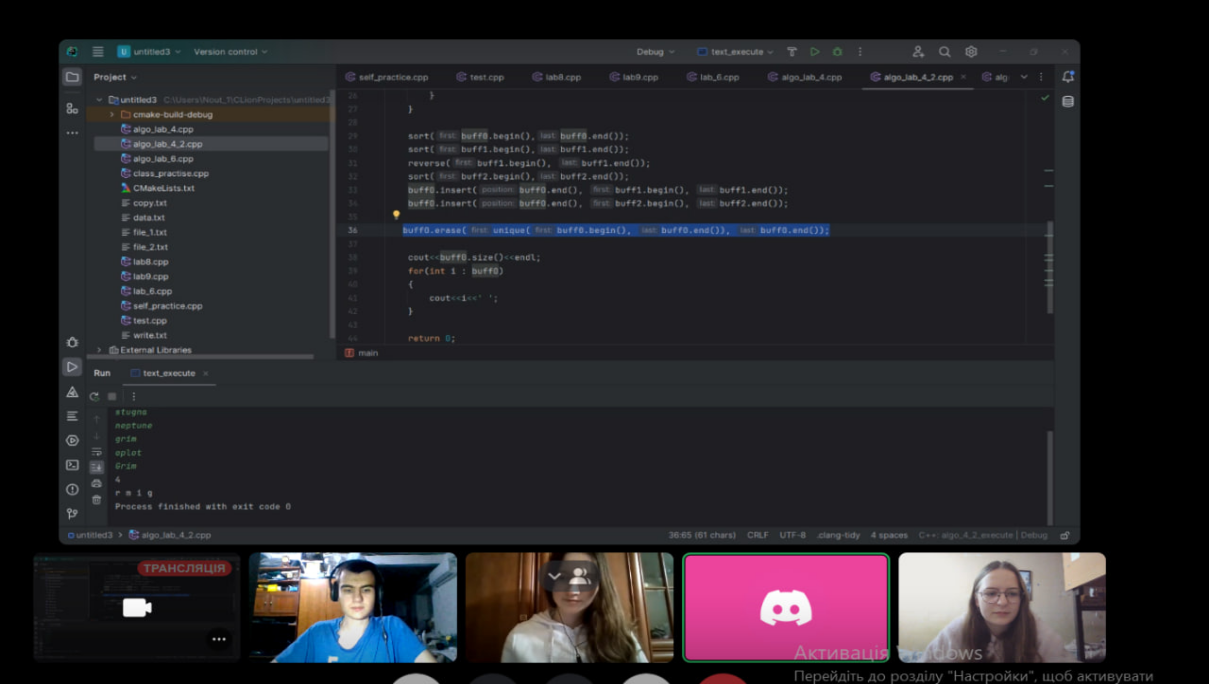


Image 5: Зустріч з командою

# **Висновки:**

Під час вивчення даної теми я опрацювала теоретичні відомісті про списки иа бінарні дерева. Виконуючи практичні та лабораторні роботи я на практиці засвоїла знання про одвозвязні та двозвязні списки та бінарні ерева реалізовуючи їх у задачах.