Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту

A blue and white logo

Description automatically generated

**Звіт**

**Звіт**

**про виконання лабораторних та практичних робіт блоку № 6**

На тему: «Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур»

**з дисципліни:** «Основи програмування»

до:

ВНС Лабораторної Роботи № 10

Алготестер Лабораторної Роботи №5

Алготестер Лабораторної Роботи № 7-8

Практичних робіт до блоку №6

**Виконав(ла):**

Студентка групи ШІ-11

Ільящук Марта Тарасівна

Львів 2024

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

**Мета роботи:** Ознайомитися з динамічними структурами: чергою, стеком, списками, деревами; навчитися працювати з ними та застосовувати різноманітні алгоритми обробки динамічних структур

**Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1. Основи динамічних структур даних
* Тема №2. Стек
* Тема №3. Черга
* Тема №4. Зв’язні списки
* Тема №5. Дерева
* Тема №6. Алгоритми обробки динамічних структур

1. Індивідуальний план опрацювання теорії:

* Тема №1. Основи динамічних структур даних

Джерела інформації: [Stack vs Heap Memory Allocation - GeeksforGeeks](https://www.geeksforgeeks.org/stack-vs-heap-memory-allocation/)

Що опрацьовано: Поняття stack, heap, прості динамічні структури

* Тема №2. Стек

Джерела інформації: [Stack Data Structure - GeeksforGeeks](https://www.geeksforgeeks.org/stack-data-structure/)

Що опрацьовано: Стек, операції над ним, приклади використання

* Тема №3. Черга

Джерела інформації: [C++ Queues](https://www.w3schools.com/cpp/cpp_queues.asp)

Що опрацьовано: Черги, їх функціонал, використання, пріоритетні черги

* Тема №4. Зв’язні списки

Джерела інформації: [Linked List in C++ - GeeksforGeeks](https://www.geeksforgeeks.org/cpp-linked-list/)

Що опрацьовано: Зв’язні списки та їх види, операції над ними, використання

* Тема №5. Дерева

Джерела інформації: [Binary Tree in C++ - GeeksforGeeks](https://www.geeksforgeeks.org/binary-tree-in-cpp/)

Що опрацьовано: Бінарні дерева, обхід дерева, основні операції, застосування

* Тема №6. Алгоритми обробки динамічних структур

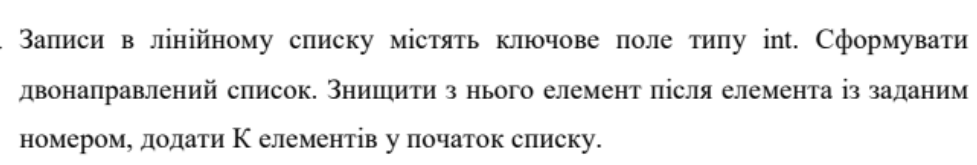
Джерела інформації: [Recursive Algorithms - GeeksforGeeks](https://www.geeksforgeeks.org/recursion-algorithms/)

Що опрацьовано: Ітеративні, рекурсивні алгоритми

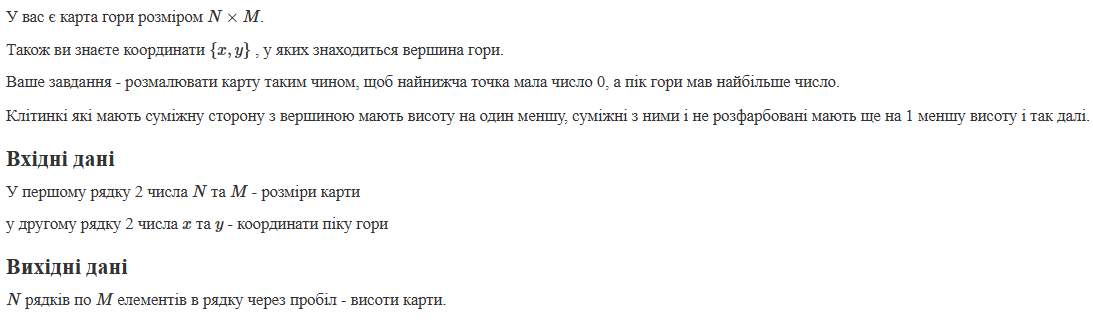
**Виконання роботи:**

1. Опрацювання завдань та вимог до середовища:

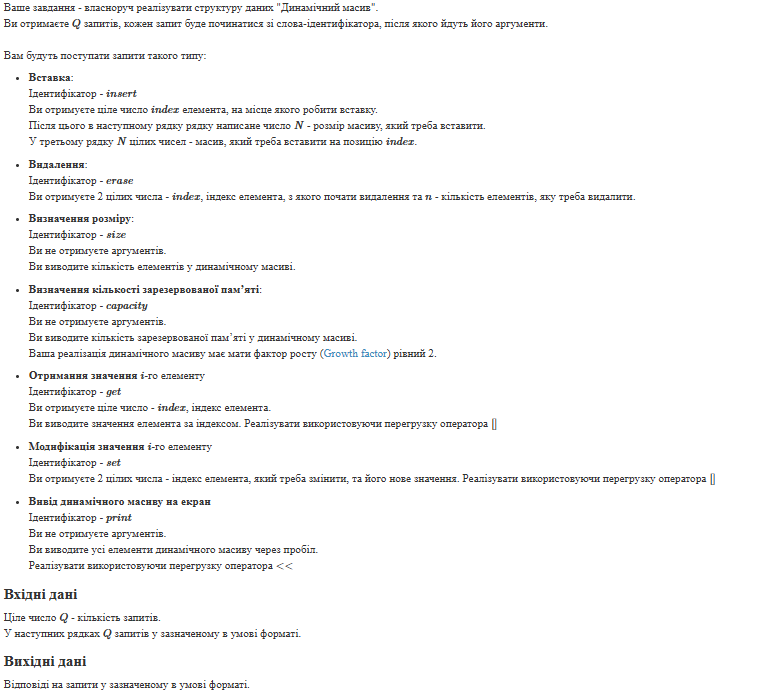
**Завдання №1. VNS lab 10 variant 8**

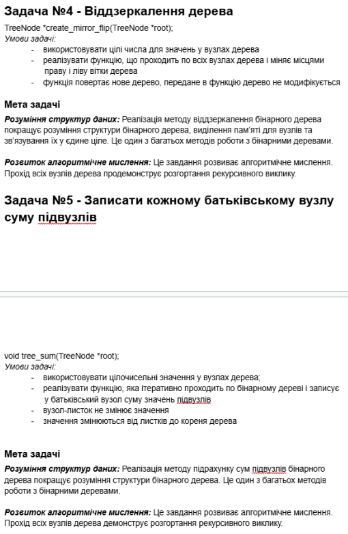
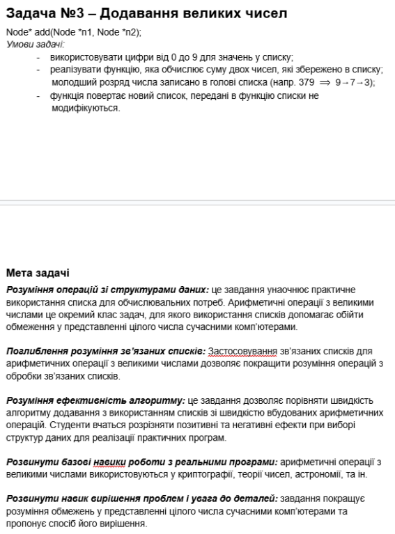
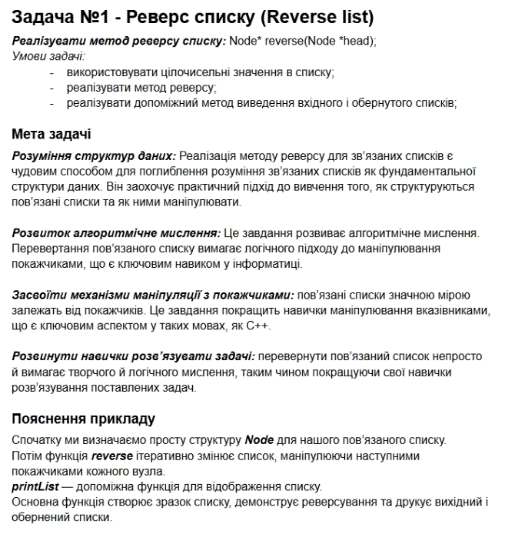
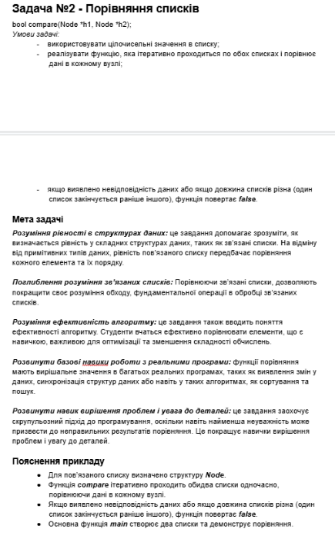


**Завдання №2. Algotester lab 5 variant 3**

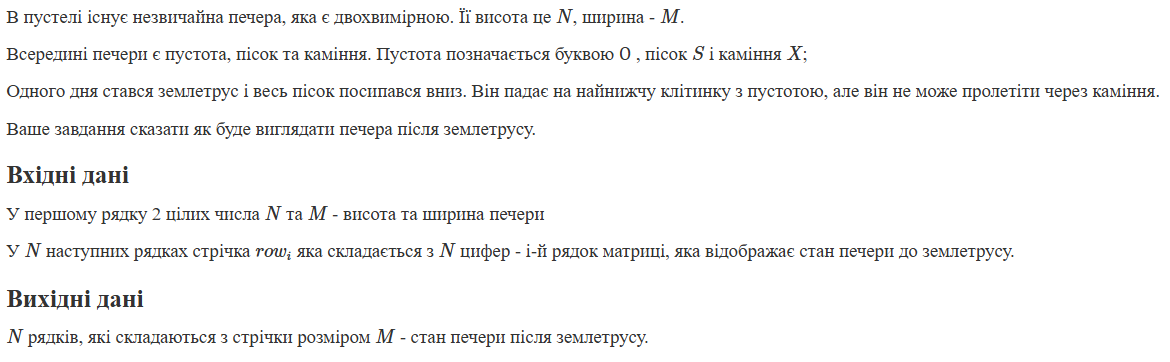
****

**Завдання №3. Algotester lab 7-8 variant 2**

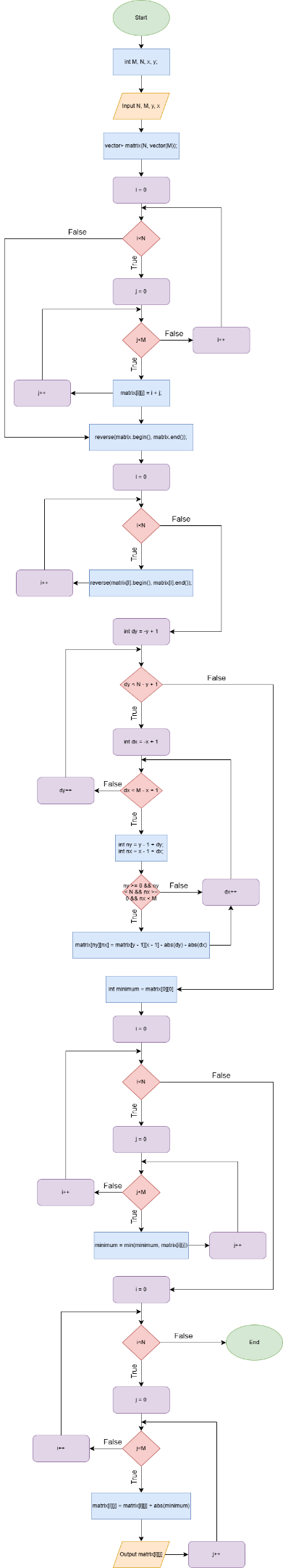
****

******Завдання №4. Class Practice Task**

**Завдання №5. Self-Practice task Algotester lab 5 variant 2**

****

1. Дизайн виконання завдань



1. Код програм з посиланням на зовнішні ресурси

[Epic 6 - Marta Iliashchuk by martailiashchuk · Pull Request #640 · artificial-intelligence-department/ai\_programming\_playground\_2024 · GitHub](https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/640)

**Завдання №1. VNS lab 10 variant 8**

#include <iostream>

using namespace std;

struct Node {

    int data;

    Node\* next;

    Node\* prev;

    Node(int value) : data(value), next(nullptr), prev(nullptr) {}

};

class DoublyLinkedList {

private:

    Node\* head;

    Node\* tail;

public:

    DoublyLinkedList() : head(nullptr), tail(nullptr) {}

    void addNodeAtStart(int value) {

        Node\* newNode = new Node(value);

        if (head == nullptr) {

            head = tail = newNode;

        } else {

            newNode->next = head;

            head->prev = newNode;

            head = newNode;

        }

    }

    void addNode(int value) {

        Node\* newNode = new Node(value);

        if (tail == nullptr) {

            head = tail = newNode;

        } else {

            tail->next = newNode;

            newNode->prev = tail;

            tail = newNode;

        }

    }

    void removeElementAfter(int value) {

        Node\* current = head;

        while (current != nullptr) {

            if (current->data == value && current->next != nullptr) {

                Node\* x = current->next;

                current->next = x->next;

                if (x->next != nullptr) {

                    x->next->prev = current;

                } else {

                    tail = current;

                }

                delete x;

                return;

            }

            current = current->next;

        }

    }

    void printList() {

        Node\* current = head;

        while (current != nullptr) {

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

            current = current->next;

        }

        cout << endl;

    }

// Free memory deconstructor

    ~DoublyLinkedList() {

        while (head != nullptr) {

            Node\* x = head;

            head = head->next;

            delete x;

        }

    }

};

int main() {

    DoublyLinkedList list;

    list.addNode(1);

    list.addNode(2);

    list.addNode(3);

    list.addNode(4);

    list.addNode(5);

    cout << "Original list: ";

    list.printList();

    list.removeElementAfter(2);

    cout << "List after removing element after 2: ";

    list.printList();

    int K = 3;

    for (int i = 0; i < K; ++i) {

        list.addNodeAtStart(10 + i);

    }

    cout << "List after adding 3 elements at the start: ";

    list.printList();

    return 0;

}

**Завдання №2. Algotester lab 5 variant 3**

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main() {

    int M, N, x, y;

    cin >> N >> M >> y >> x;

    vector<vector<int>> matrix(N, vector<int>(M));

    for (int i = 0; i < N; i++) {

        for (int j = 0; j < M; j++) {

            matrix[i][j] = i + j;

        }

    }

    reverse(matrix.begin(), matrix.end());

    for (int i = 0; i < N; i++) {

        reverse(matrix[i].begin(), matrix[i].end());

    }

    for (int dy = -y + 1; dy < N - y + 1; dy++) {

        for (int dx = -x + 1; dx < M - x + 1; dx++) {

            int ny = y - 1 + dy;

            int nx = x - 1 + dx;

            if (ny >= 0 && ny < N && nx >= 0 && nx < M) {

                matrix[ny][nx] = matrix[y - 1][x - 1] - abs(dy) - abs(dx);

            }

        }

    }

    int minimum = matrix[0][0];

    for (int i = 0; i < N; i++) {

        for (int j = 0; j < M; j++) {

            minimum = min(minimum, matrix[i][j]);

        }

    }

    for (int i = 0; i < N; i++) {

        for (int j = 0; j < M; j++) {

            matrix[i][j] = matrix[i][j] + abs(minimum);

            cout << matrix[i][j] << " ";

        }

        cout << endl;

    }

    return 0;

}

**Завдання №3. Algotester lab 7-8 variant 2**

#include <iostream>

using namespace std;

template <class T>

class DynamicArray {

private:

    T \*data;

//Growth factor 2

    void ResizeIfNeeded(int N) {

        while (size + N >= capacity) {

            capacity \*= 2;

        }

    }

public:

    int size;

    int capacity;

    DynamicArray() {

        size = 0;

        capacity = 1;

        data = new T[1];

    }

    void insert(int index, int N, T \*elements) {

        ResizeIfNeeded(N);

// Create an array with all previous elements[i<index], add new elements starting with index, add the rest data, index of the last added element - start point

        T \*temporary = new T[capacity];

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

            temporary[i] = data[i];

        }

        for (int i = 0; i < N; i++) {

            temporary[index + i] = elements[i];

        }

        for (int i = index; i < size; i++) {

            temporary[i + N] = data[i];

        }

        size += N;

        delete[] data;

        data = temporary;

    }

    void erase(int index, int N) {

        T \*temporary = new T[capacity];

        int newSize = 0;

//Delete imitation(rebuild data without element[index])

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

            if (i < index || i >= index + N) {

                temporary[newSize] = data[i];

                newSize++;

            }

        }

        size -= N;

        delete[] data;

        data = temporary;

    }

    int Size() const {

        return size;

    }

    int Capacity() const {

        return capacity;

    }

    T get(int index) const {

        return data[index];

    }

//Change element[index]

    void set(int index, T value) {

        data[index] = value;

    }

    void print(const string separator = " ") const {

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

            cout << data[i];

            if (i < size - 1) {

                cout << separator;

            }

        }

        cout << endl;

    }

};

int main() {

//data (typename)

    DynamicArray<int> array;

    int Q;

    cin >> Q;

    while (Q--) {

        string command;

        cin >> command;

        if (command == "insert") {

            int index, N;

            cin >> index >> N;

            int \*elements = new int[N];

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

                cin >> elements[i];

            }

            array.insert(index, N, elements);

            delete[] elements;

        }

        else if (command == "erase") {

            int index, N;

            cin >> index >> N;

            array.erase(index, N);

        }

        else if (command == "size") {

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

        }

        else if (command == "capacity") {

            cout << array.Capacity() << endl;

        }

        else if (command == "get") {

            int index;

            cin >> index;

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

        }

        else if (command == "set") {

            int index, N;

            cin >> index >> N;

            array.set(index, N);

        }

        else if (command == "print") {

            array.print();

        }

    }

    return 0;

}

**Завдання №4. Class Practice Task**

#include <iostream>

using namespace std;

struct Node {

    int nodeValue;

    Node\* nextNodeValue;

    Node(int value) : nodeValue(value), nextNodeValue(nullptr) {}

};

// Task 1: Reversing linked list

Node\* reverse(Node\*& head) {

    Node\* prev = nullptr;

    Node\* current = head;

    Node\* next = nullptr;

    while (current != nullptr) {

        next = current->nextNodeValue;

        current->nextNodeValue = prev; // 1 -> nullptr, next - current...4 - >... 2 ->...

        prev = current;

        current = next;

    }

    return prev;

}

// Task 2: Comparing two linked lists

// Going through all elements while element = element

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

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

        if (h1->nodeValue != h2->nodeValue) {

            return false;

        }

        h1 = h1->nextNodeValue;

        h2 = h2->nextNodeValue;

    }

    return h1 == nullptr && h2 == nullptr;

}

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

    Node\* newNode = new Node(value);

    if (head == nullptr) {

        head = newNode;

    } else {

        Node\* x = head;

        while (x->nextNodeValue != nullptr) {

            x = x->nextNodeValue;

        }

        x->nextNodeValue = newNode;

    }

}

// Task 3: Adding two linked lists

Node\* add(Node\* n1, Node\* n2) {

    Node\* result = nullptr;

    Node\* x = nullptr;

    int carry = 0;

    while (n1 != nullptr || n2 != nullptr || carry != 0) {

        int sum = carry;

        if (n1 != nullptr) {

            sum += n1->nodeValue;

            n1 = n1->nextNodeValue;

        }

        if (n2 != nullptr) {

            sum += n2->nodeValue;

            n2 = n2->nextNodeValue;

        }

        carry = sum / 10;

        int digit = sum % 10;

        addToList(result, digit);

    }

    return result;

}

void printList(Node\* head) {

    while (head != nullptr) {

        cout << head->nodeValue << " ";

        head = head->nextNodeValue;

    }

    cout << endl;

}

void printElements(Node\* head) {

    head = reverse(head);

    while (head != nullptr) {

        cout << head->nodeValue;

        head = head->nextNodeValue;

    }

    cout << endl;

}

// Free memory

void clearList(Node\*& head) {

    while (head != nullptr) {

        Node\* temp = head;

        head = head->nextNodeValue;

        delete temp;

    }

}

int main() {

    Node\* head = nullptr;

    addToList(head, 1);

    addToList(head, 2);

    addToList(head, 3);

    addToList(head, 4);

    cout << "Task 1" << endl;

    printList(head);

    head = reverse(head);

    cout << "Reversed list: " << endl;

    printList(head);

    Node\* h1 = nullptr;

    addToList(h1, 1);

    addToList(h1, 3);

    addToList(h1, 7);

    addToList(h1, 5);

    cout << endl << "Task 2: " << endl;

    cout << "First list: " << endl;

    printList(h1);

    Node\* h2 = nullptr;

    addToList(h2, 1);

    addToList(h2, 4);

    addToList(h2, 7);

    cout << "Second list: " << endl;

    printList(h2);

    if (compare(h1, h2)) {

        cout << "=" << endl;

    } else {

        cout << "!=" << endl;

    }

    cout << endl << "Task 3: " << endl;

    Node\* result = add(h1, h2);

    cout << "Number: ";

    printElements(h1);

    cout << "Next number: ";

    printElements(h2);

    cout << "Sum: ";

    printElements(result);

// Free memory

    clearList(head);

    clearList(h1);

    clearList(h2);

    clearList(result);

    return 0;

}

#include <iostream>

using namespace std;

struct TreeNode {

    int data;

    TreeNode\* left;

    TreeNode\* right;

    TreeNode(int value): data(value), left(nullptr), right(nullptr) {}

};

void printTree(TreeNode\* root) {

    if (root != nullptr) {

        cout << root->data << " ";

        printTree(root->left);

        printTree(root->right);

    }

}

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

    if (root == nullptr) {

        return nullptr;

    }

    TreeNode\* left = create\_mirror\_flip(root->left);

    TreeNode\* right = create\_mirror\_flip(root->right);

    root->left = right;

    root->right = left;

    return root;

}

int tree\_sum(TreeNode\* root) {

    if (root == nullptr) {

        return 0;

    }

    int leftSum = tree\_sum(root->left);

    int rightSum = tree\_sum(root->right);

    if (root->left != nullptr || root->right != nullptr) {

        root->data += leftSum + rightSum;

    }

    return root->data;

}

//Free memory

void deleteTree(TreeNode\* root) {

    if (root == nullptr) {

        return;

    }

// Delete left and right subtrees

    deleteTree(root->left);

    deleteTree(root->right);

// Delete current node

    delete root;

}

int main() {

    TreeNode\* root = new TreeNode(1);

    root->left = new TreeNode(2);

    root->right = new TreeNode(4);

    root->left->left = new TreeNode(8);

    root->right->left = new TreeNode(9);

    root->right->right = new TreeNode(7);

    cout << "Task 4: " << endl;

    printTree(root);

    root = create\_mirror\_flip(root);

    cout << endl << "Mirrored tree: " << endl;

    printTree(root);

    cout << endl << "Task 5: " << endl;

    tree\_sum(root);

    cout << "Tree sum: ";

    printTree(root);

    cout << endl;

// Free memory

    deleteTree(root);

    return 0;

}

**Завдання №5. Self-Practice task Algotester lab 5 variant 2**

#include <iostream>

#include <vector>

using namespace std;

int main() {

    int N, M;

    cin >> N >> M;

    vector<vector<char>> rows(N, vector<char>(M));

    for (int i = 0; i < N; i++) {

        string row;

        cin >> row;

        for (int j = 0; j < M; j++) {

            rows[i][j] = row[j];

        }

    }

    for (int j = 0; j < M; j++) {

        for (int i = N - 2; i >= 0; i--) {

            if (rows[i][j] == 'S') {

                int k = i;

                while (k + 1 < N && rows[k + 1][j] == 'O') {

                    k++;

                }

                if (k != i && rows[k][j] != 'X') {

                    rows[k][j] = 'S';

                    rows[i][j] = 'O';

                }

            }

        }

    }

    for (int i = 0; i < N; i++) {

        for (int j = 0; j < M; j++) {

            cout << rows[i][j];

        }

        cout << endl;

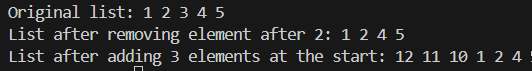
    }

    return 0;

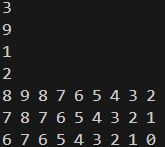
}

4.Результати виконання завдань

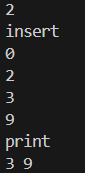
**Завдання №1. VNS lab 10 variant 8**

****

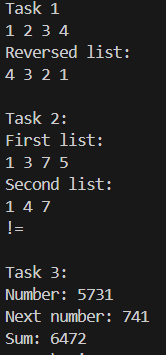
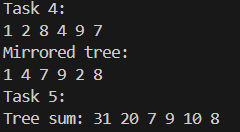
**Завдання №2. Algotester lab 5 variant 3**

****

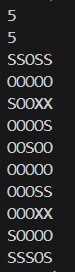
**Завдання №3. Algotester lab 7-8 variant 2**

****

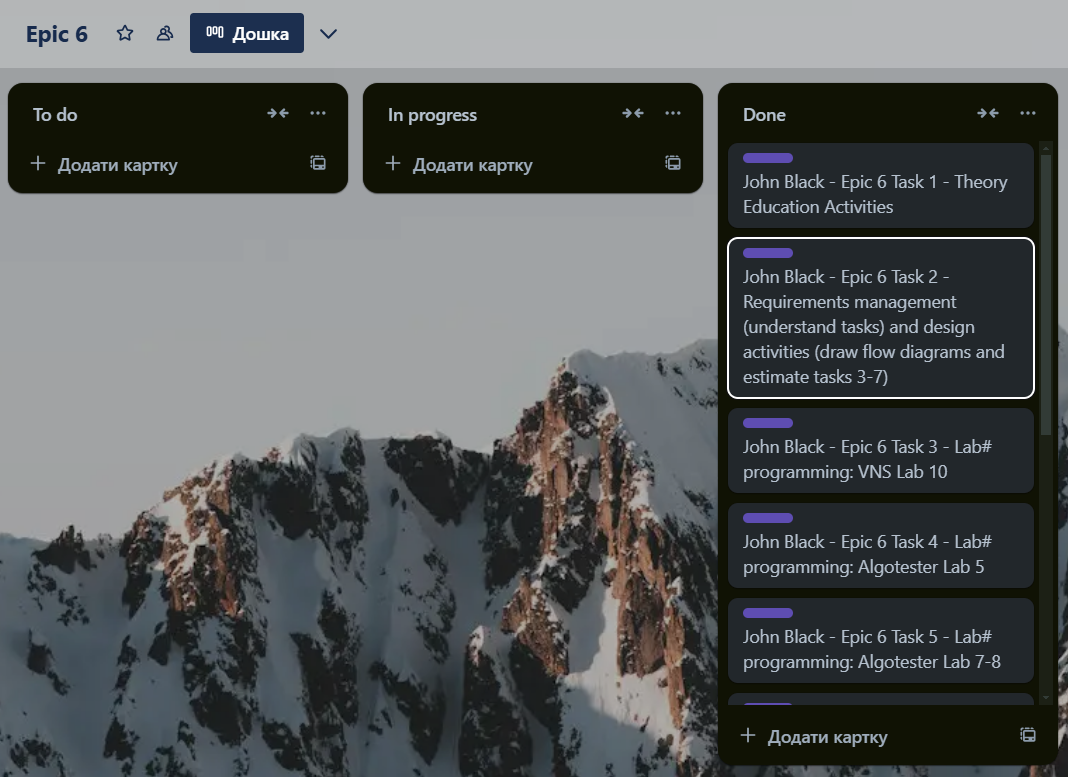
**Завдання №4. Class Practice Task**

****

**Завдання №5. Self-Practice task Algotester lab 5 variant 2**

****

5. Кооперація з командою



**Висновок:** Під час виконання роботи я ознайомилася з динамічними структурами та навчилася використовувати їх на практиці.