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»

***Виконав:***

студент групи ШІ-13

Сіренко Юрій Сергійович

# **Тема роботи:**

Динамічні структури. Їх типи, властивості, алгоритми обробки.

# **Мета роботи:**

Метою даної роботи є розуміння динамічних структур даних, проектування та реалізація структур даних, вивчення базових операцій, що виконуються над динамічними структурами (додавання, видалення, пошук, сортування), застосування динамічних структур у практичних задачах.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Динамічний масив.
* Тема №2: Бінарне дерево.
* Тема №3: Однозв’язний та двозв’язний списки.

1. Індивідуальний план опрацювання теорії:

* Тема №1: Динамічний масив.
  + Джерела Інформації
    - Відео - <https://www.youtube.com/watch?v=GHLv3fNLOh0>
    - Стаття 1 - [How to create a dynamic array of integers in c++](https://stackoverflow.com/questions/4029870/how-to-create-a-dynamic-array-of-integers-in-c)
    - Стаття 2 - [How do dynamic arrays work?](https://www.geeksforgeeks.org/how-do-dynamic-arrays-work/)
    - Курс - Beginning C++ Programming - From Beginner to Beyond
  + Що опрацьовано:
    - Ознайомлення із Специфікацією Динамічного Масиву:
    - Розуміння основних властивостей динамічного масиву.
    - Знання процесу його створення та розширення в пам'яті за необхідності.
    - Операції Запису та Читання:
    - Вивчення процесу зчитування та запису даних у динамічний масив.
    - Розуміння індексації елементів в масиві та його взаємодії з пам'яттю.
    - Маніпулювання Розміром Масиву:
    - Освоєння способів зміни розміру динамічного масиву.
    - Розгляд ситуацій, коли потрібно збільшити чи зменшити розмір масиву.
    - Керування Пам'яттю:
    - Ознайомлення із процесом управління пам'яттю для динамічного масиву.
    - Розуміння важливості вивільнення пам'яті після завершення роботи.
    - Ефективне Використання Динамічного Масиву:
  + Статус: Ознайомлений
  + Початок опрацювання теми: 07.12.2023
  + Звершення опрацювання теми: 08.12.2023
* Тема №2: Бінарне дерево.
  + Джерела Інформації:
    - Відео - <https://www.youtube.com/watch?v=zuuAPYiMYDA>
    - Стаття 1 - [Introduction to binary tree data structure and algorithm](https://www.geeksforgeeks.org/introduction-to-binary-tree-data-structure-and-algorithm-tutorials/).
    - Стаття 2 - <https://www.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion/>
    - Курс - Beginning C++ Programming - From Beginner to Beyond
  + Що опрацьовано:
    - Розуміння того, що бінарне дерево - це ієрархічна структура даних, в якій кожен вузол має не більше двох нащадків (лівого та правого).
    - Операції на бінарному дерева
    - Розуміння властивостей BST, де для кожного вузла всі значення в лівому піддереві менше за його значення, а всі значення в правому піддереві більше.
    - Алгоритми для бінарного дерева:
    - Вивчення того, як бінарні дерева використовуються в різних областях програмування, таких як бази даних, обробка виразів, компілятори, графічні системи тощо.
    - Реалізація та оптимізація:
    - Реалізація власних класів або структур для представлення бінарного дерева в коді.
    - Оптимізація операцій та алгоритмів для досягнення більшої ефективності та швидкодії.
    - Розуміння можливих проблем та виправлення помилок.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 07.12.2023
  + Звершення опрацювання теми: 08.12.2023
* Тема №3: Однозв’язний та двозв’язний списки.
  + Джерела Інформації:
    - Відео - <https://www.youtube.com/watch?v=RCHGco2NvMk> .
    - Стаття 1 - [https://www.geeksforgeeks.org/difference-between-singly-linked-list-and-doubly-linked-list/.](https://www.geeksforgeeks.org/difference-between-singly-linked-list-and-doubly-linked-list/)
    - Стаття 2 - <https://www.geeksforgeeks.org/data-structures/linked-list/doubly-linked-list/>
    - Курс - Beginning C++ Programming - From Beginner to Beyond
  + Що опрацьовано:
    - Однозв’язний список:
    - Структура вузла - Елементи однозв'язного списку складаються з вузлів, кожен із яких містить дані та вказівник на наступний вузол у списку.
    - Вставка - Додавання нового вузла на початок або в кінець списку.
    - - Вставка в середину списку на певну позицію.
    - Видалення - Видалення вузла з початку, кінця або з середини списку.
    - Пошук - Пошук вузла за значенням або за індексом.
    - Перебір - Проходження через всі вузли списку для операцій обробки.
    - Двозв'язний Список:
    - Структура вузла - Елементи однозв'язного списку складаються з вузлів, кожен із яких містить дані та вказівник на наступний вузол у списку.
    - Вставка - Додавання нового вузла на початок або в кінець списку.
    - - Вставка в середину списку на певну позицію.
    - Видалення - Видалення вузла з початку, кінця або з середини списку.
    - Пошук - Пошук вузла за значенням або за індексом.
    - Перебір - Проходження через всі вузли списку для операцій обробки.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 07.12.2023
  + Звершення опрацювання теми: 08.12.2023

# **Виконання роботи:**

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

Завдання №1 Опрацювання теорії

* Деталі завдання: опрацювання вище перечисленої теорії. Розбір статей, перегляд відео та імплементація теорії на практиці. Вивчення найосновніших тем.

Завдання №2 VNS Lab 10

* Варіант завдання – 1

Записи в лінійному списку містять ключове поле типу int. Сформувати

однонаправлений список. Знищити з нього елемент із заданим номером,

додати елемент із заданим номером;

* Деталі завдання

Написати програму, у якій створюються динамічні структури й виконати

їхню обробку у відповідності зі своїм варіантом.

Для кожного варіанту розробити такі функції:

1. Створення списку.

2. Додавання елемента в список (у відповідності зі своїм варіантом).

3. Знищення елемента зі списку (у відповідності зі своїм варіантом).

4. Друк списку.

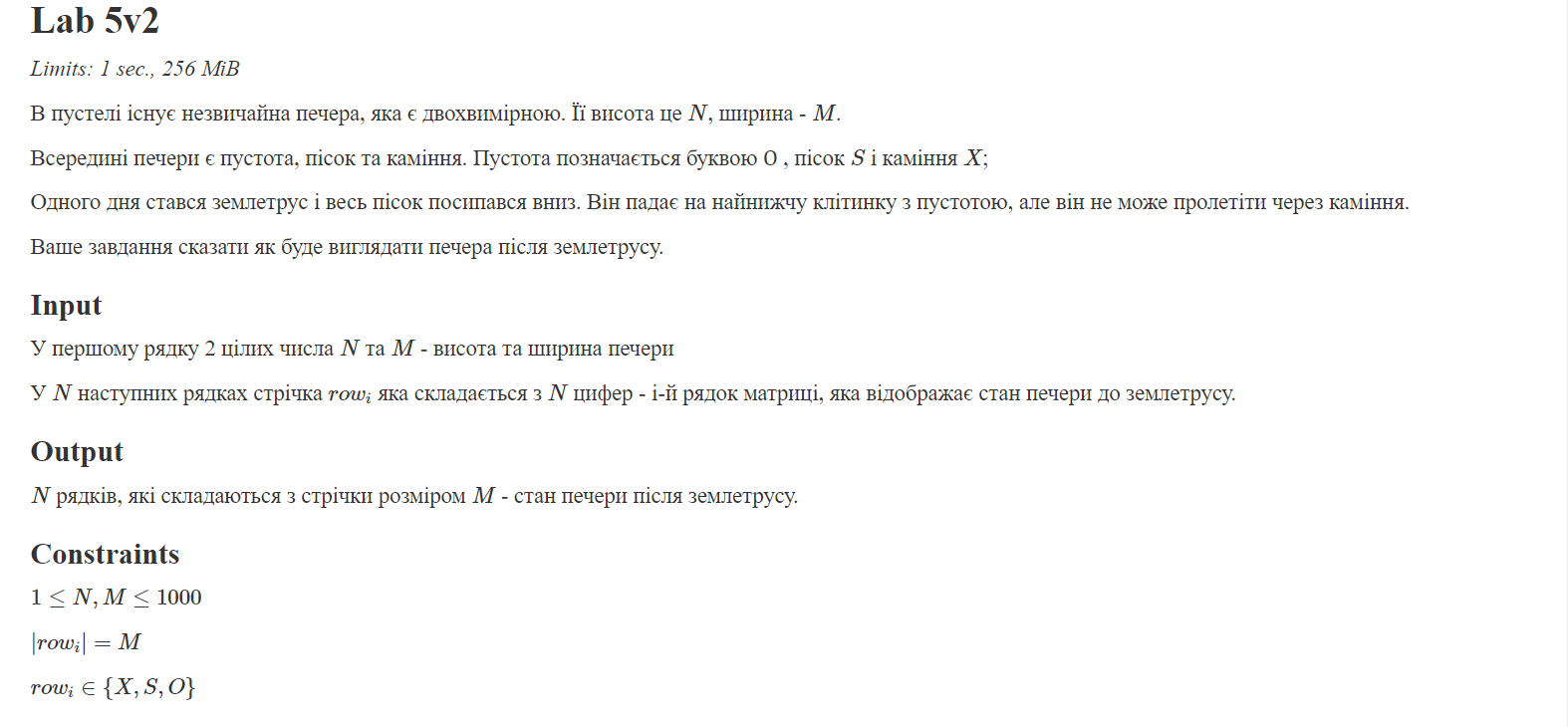
5. Запис списку у файл.

6. Знищення списку.

7. Відновлення списку з файлу.

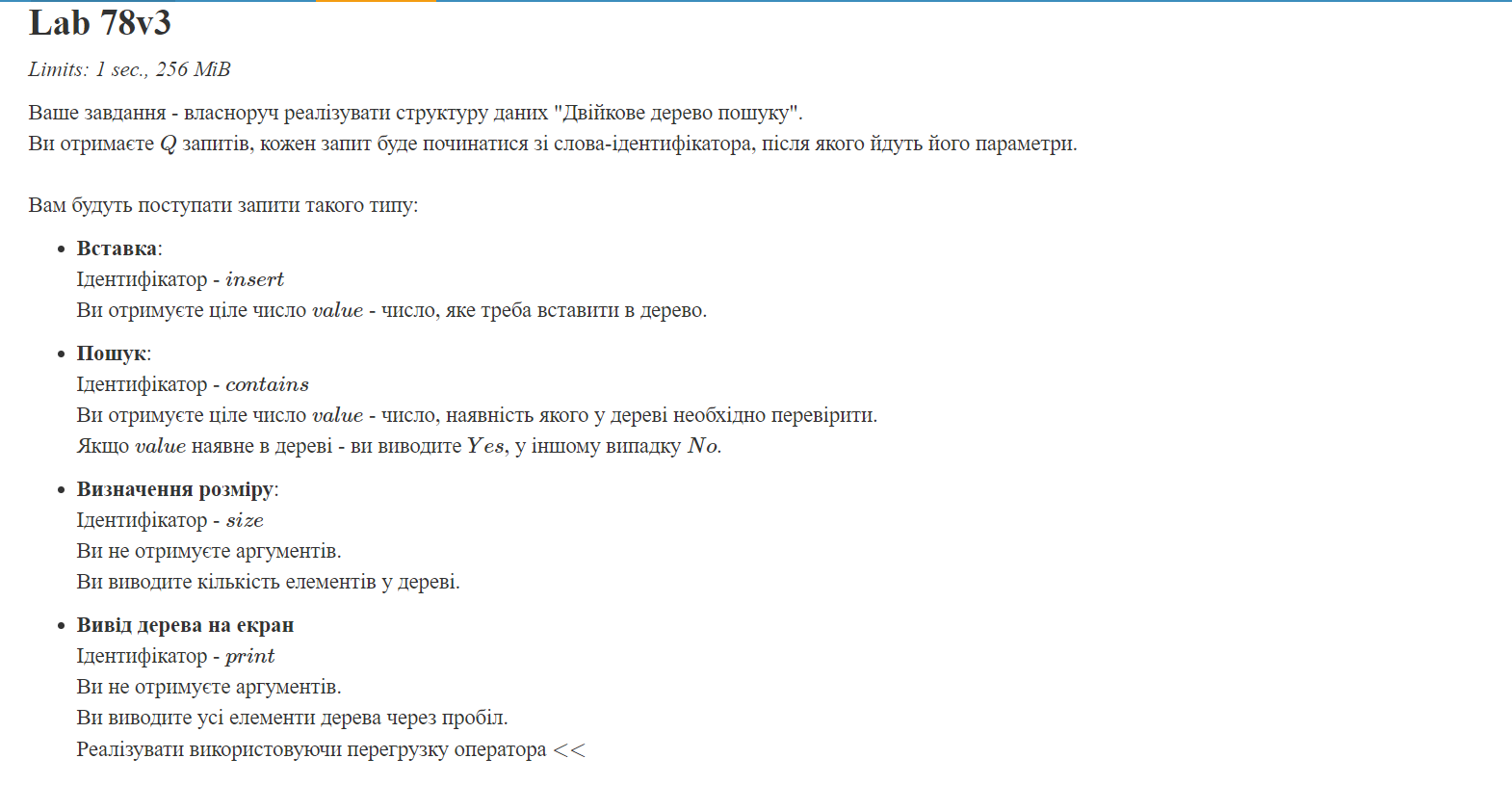
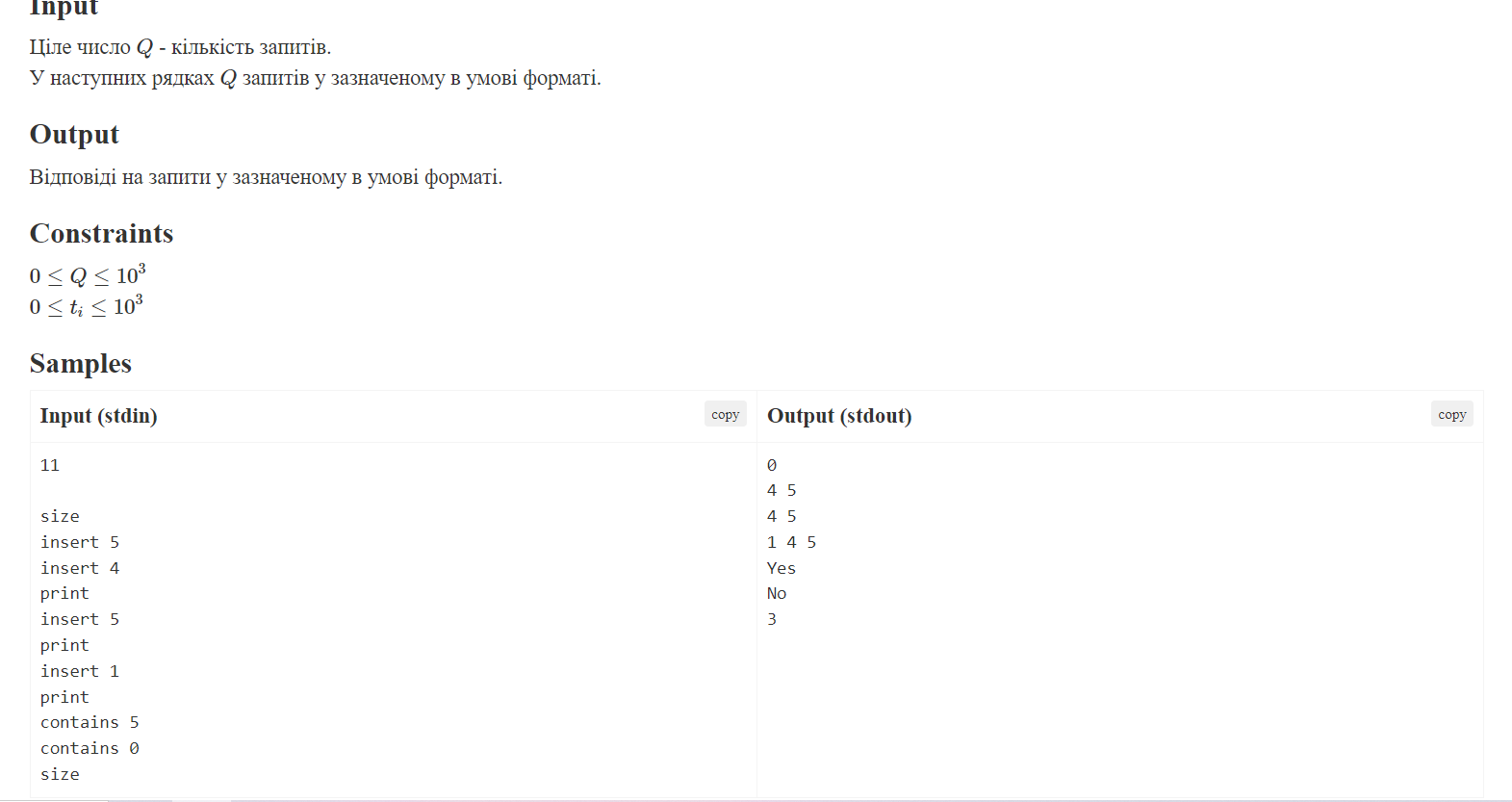
Завдання №3 Algotester Lab 5

* Варіант завдання - 2
* Деталі завдання



Завдання №4 Algotester Lab 78

* Варіант завдання - 3
* Деталі завдання

Завдання №5 Class Practice 1

* Деталі завдання

## Задача №1 - Реверс списку (Reverse list)

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

*Умови задачі:*

-       використовувати цілочисельні значення в списку;

-       реалізувати метод реверсу;

-       реалізувати допоміжний метод виведення вхідного і обернутого списків;

Завдання №6 Class Practice 2

* Деталі завдання

## Задача №2 - Порівняння списків

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

*Умови задачі:*

-       використовувати цілочисельні значення в списку;

-       реалізувати функцію, яка ітеративно проходиться по обох списках і порівнює дані в кожному вузлі;

-       якщо виявлено невідповідність даних або якщо довжина списків різна (один список закінчується раніше іншого), функція повертає ***false***.

Завдання №7 Class Practice 3

* Деталі завдання

## Задача №3 – Додавання великих чисел

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

*Умови задачі:*

-       використовувати цифри від 0 до 9 для значень у списку;

-       реалізувати функцію, яка обчислює суму двох чисел, які збережено в списку; молодший розряд числа записано в голові списка (напр. 379  ⟹  9→7→3);

-       функція повертає новий список, передані в функцію списки не модифікуються.

Завдання №8 Class Practice 4

* Деталі завдання

## Задача №4 - Віддзеркалення дерева

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

*Умови задачі:*

-       використовувати цілі числа для значень у вузлах дерева

-       реалізувати функцію, що проходить по всіх вузлах дерева і міняє місцями праву і ліву вітки дерева

-       функція повертає нове дерево, передане в функцію дерево не модифікується

Завдання №9 Class Practice 5

* Деталі завдання

## Задача №5 - Записати кожному батьківському вузлу суму підвузлів

void tree\_sum(TreeNode \*root);

*Умови задачі:*

-       використовувати цілочисельні значення у вузлах дерева;

-       реалізувати функцію, яка ітеративно проходить по бінарному дереві і записує у батьківський вузол суму значень підвузлів

-       вузол-листок не змінює значення

-       значення змінюються від листків до кореня дерева

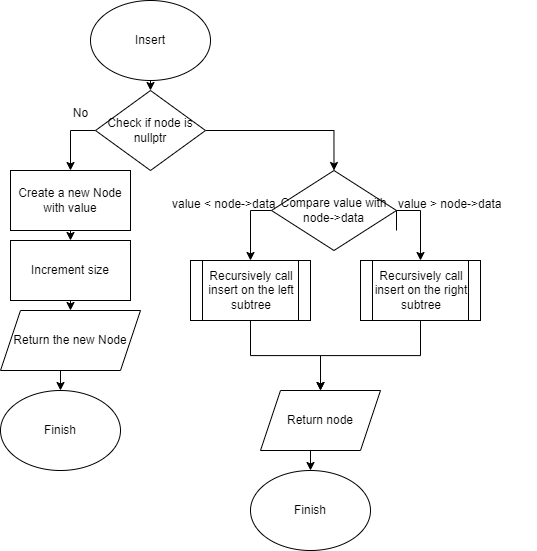
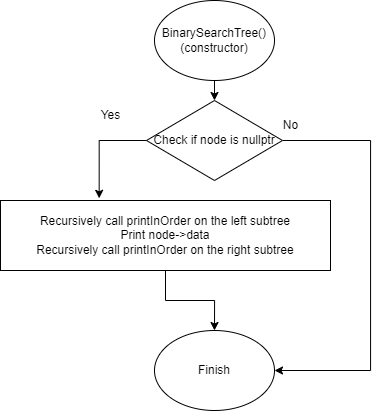
## **2. Дизайн та планована оцінка часу виконання завдань:**

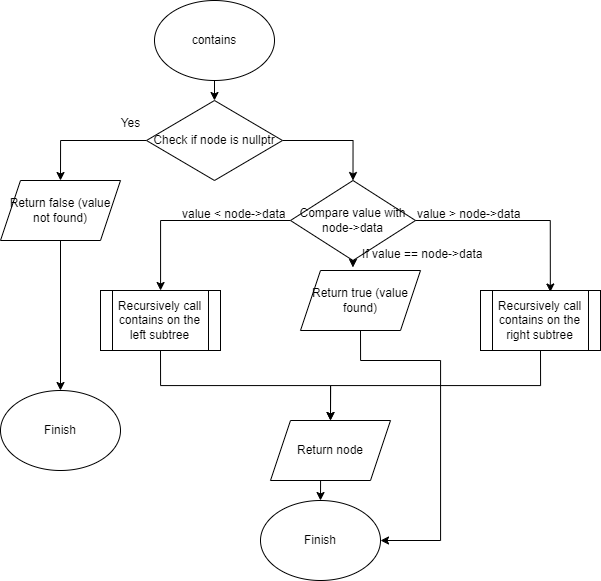
Програма №1 VNS Lab 10

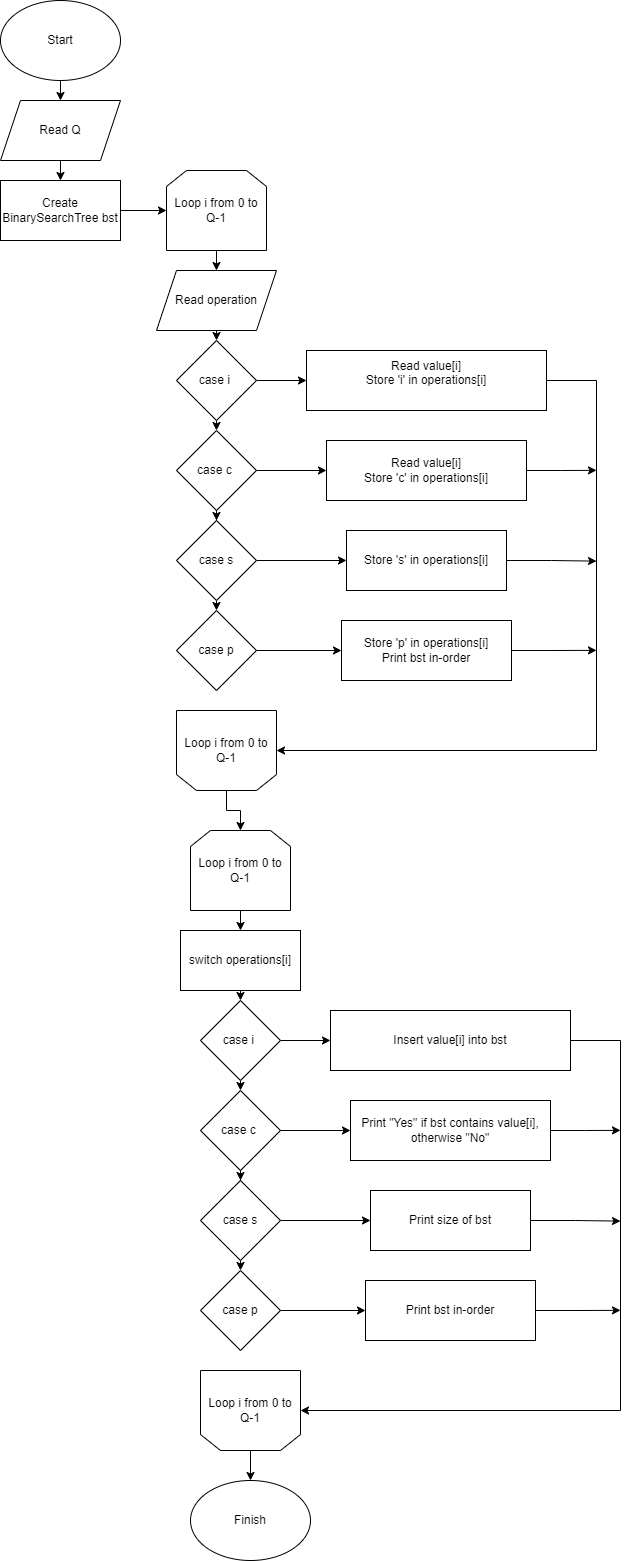
* Блок-схема - відсутня
* Планований час на реалізацію - 1 година
* Час затрачений на виконання завдання – 2 години

Програма №2 Algotester Lab 5

* Блок-схема







* Планований час на реалізацію – 1 година
* Час затрачений на виконання завдання – 2 години

Програма №3 Algotester Lab 78

* Блок-схема - відсутня
* Планований час на реалізацію - 1 година
* Час затрачений на виконання завдання – 4 години

Програма №4 Class Practice 1

* Блок-схема - відсутня
* Планований час на реалізацію - 1 година
* Час затрачений на виконання завдання – 30 хвилин

Програма №5 Class Practice 2

* Блок-схема - відсутня
* Планований час на реалізацію – 1 година
* Час затрачений на виконання завдання – 30 хвилин

Програма №6 Class Practice 3

* Блок-схема - відсутня
* Планований час на реалізацію – 1 година
* Час затрачений на виконання завдання – 30 хвилин

Програма №7 Class Practice 4

* Блок-схема - відсутня
* Планований час на реалізацію – 1 година
* Час затрачений на виконання завдання – 30 хвилин

Програма №8 Class Practice 5

* Блок-схема - відсутня
* Планований час на реалізацію – 1 година
* Час затрачений на виконання завдання – 30 хвилин

## **3. 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/pull/751/files#diff-dfa9cbd217262ad24a7dba72c60a7ea3ea7d0b1483eeffa14c35ca1d7385c176>

#include <iostream>

#include <fstream>

using namespace std;

struct Node

{

    int data;

    Node \*link;

};

typedef Node\* nodeptr;

nodeptr createList()

{

    return nullptr;

}

void printList(nodeptr head)

{

    nodeptr current = head;

    if(current==nullptr)

    {

        cout<<"List is empty";

    }

    while (current != nullptr) {

        cout << current->data << " ";

        current = current->link;

    }

    cout << endl;

}

void insert(nodeptr& head,int data)

{

    nodeptr tempPtr;

    tempPtr = new Node;

    tempPtr->data = data;

    tempPtr->link = head;

    head = tempPtr;

}

void writeToFile(nodeptr head, const string& filename)

{

    ofstream outFile(filename);

    if (!outFile) {

        cerr << "Unable to open the file: " << filename << endl;

        return;

    }

    nodeptr current = head;

    while (current != nullptr) {

        outFile << current->data << " ";

        current = current->link;

    }

    outFile.close();

}

nodeptr restoreFromFile(const string& filename)

{

    ifstream inFile(filename);

    if (!inFile)

    {

        cerr << "Unable to open the file: " << filename << endl;

        return nullptr;

    }

    nodeptr head = createList();

    int data;

    while (inFile >> data)

    {

        insert(head, data);

    }

    inFile.close();

    return head;

}

void deleteList(nodeptr& head)

{

    while (head != nullptr)

    {

        nodeptr temp = head;

        head = head->link;

        delete temp;

    }

}

void addNode(nodeptr& head, int data, int position)

{

    nodeptr newNode = new Node;

    newNode->data = data;

    if (position == 1 || head == nullptr)

    {

        newNode->link = head;

        head = newNode;

    }

    else

    {

        nodeptr temp = head;

        for (int i = 1; i < position - 1 && temp != nullptr; ++i)

            temp = temp->link;

        if (temp != nullptr)

        {

            newNode->link = temp->link;

            temp->link = newNode;

        }

        else

        {

            cout << "Invalid position to insert." << endl;

        }

    }

}

void deleteNode(Node \*\*head, int position)

{

    if (\*head == nullptr)

    {

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

        return;

    }

    Node \*current = \*head;

    Node \*previous = nullptr;

    if (position == 1)

    {

        \*head = current->link;

        delete current;

        current = nullptr;

    }

    else

    {

        for (int i = 1; i < position && current != nullptr; ++i)

        {

            previous = current;

            current = current->link;

        }

        if (current == nullptr)

        {

            cout << "Invalid position to delete." << endl;

            return;

        }

        previous->link = current->link;

        delete current;

        current = nullptr;

    }

}

int main()

{

    int pos\_to\_add,number\_to\_add,pos\_to\_delete;

    cout<<"Enter pos to add: ";

    cin>>pos\_to\_add;

    cout<<"Enter number to add: ";

    cin>>number\_to\_add;

    cout<<"Enter pos to delete: ";

    cin>>pos\_to\_delete;

    nodeptr head = createList();

    insert(head,228);

    insert(head,17);

    insert(head,35);

    insert(head,44);

    printList(head);

     addNode(head,number\_to\_add,pos\_to\_add);

    printList(head);

     deleteNode(&head,pos\_to\_delete);

    printList(head);

    writeToFile(head,"test.txt");

    deleteList(head);

    printList(head);

    head=restoreFromFile("test.txt");

    printList(head);

}

*Код програми №1 VNS Lab 5*

Завдання №2 Algotester Lab 5

Посилання на файл програми у пул-запиті GitHub

[https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/751/files#diff-b6b6eeb8c63000df79d2a5c2afecd7bf8c7a97d810cfe794dfa9cf3f41abb50a](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/751/files" \l "diff-b6b6eeb8c63000df79d2a5c2afecd7bf8c7a97d810cfe794dfa9cf3f41abb50a)

#include <iostream>

#include <string>

using namespace std;

int main()

{

    int n,m;

    cin>>n>>m;

    char cave[n][m];

    for(int t = 0;t<n;t++)

    {

        for(int i = 0;i<m;i++)

        {

            cin>>cave[t][i];

        }

    }

    cout<<"\n";

    for(int i = 0;i<m;i++)

    {

        for(int t=n-1;t!=-1;t--)

        {

            if(cave[t][i]=='S')

            {

                if(cave[t+1][i]=='O')

                {

                    cave[t+1][i]='S';

                    cave[t][i]='O';

                    t=n-1;

                }

            }

        }

    }

    for(int t = 0;t<n;t++)

    {

        for(int i = 0;i<m;i++)

        {

            cout<<cave[t][i];

        }

        cout<<"\n";

    }

}

*Код програми №2 Algotester Lab 5*

Завдання №3 Algotester Lab 78

Посилання на файл програми у пул-запиті GitHub

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/751/files#diff-ff141672b236676a84b77964e855b1c13a63d4968ffd85c17a409a9df1168c9e>

#include <iostream>

#include <vector>

using namespace std;

template <typename T>

class BinarySearchTree

{

private:

    struct Node {

        T data;

        Node\* left;

        Node\* right;

        Node(const T& value) : data(value), left(nullptr), right(nullptr) {}

    };

    Node\* root;

    int size;

    Node\* insert(Node\* node, const T& value) {

        if (node == nullptr) {

            size++;

            return new Node(value);

        }

        if (value < node->data) {

            node->left = insert(node->left, value);

        } else if (value > node->data) {

            node->right = insert(node->right, value);

        }

        return node;

    }

    bool contains(Node\* node, const T& value) const {

        if (node == nullptr) {

            return false;

        }

        if (value == node->data) {

            return true;

        } else if (value < node->data) {

            return contains(node->left, value);

        } else {

            return contains(node->right, value);

        }

    }

    void printInOrder(Node\* node) const {

        if (node != nullptr) {

            printInOrder(node->left);

            std::cout << node->data << " ";

            printInOrder(node->right);

        }

    }

public:

    BinarySearchTree() : root(nullptr), size(0) {}

    void insert(const T& value) {

        root = insert(root, value);

    }

    bool contains(const T& value) const {

        return contains(root, value);

    }

    int getSize() const {

        return size;

    }

    void print() const {

        printInOrder(root);

        std::cout << std::endl;

    }

};

int main()

{

    int Q;

    cin >> Q;

    string operation;

    char operations[Q];

    int value[Q];

    BinarySearchTree <int> bst;

    for(int i = 0;i<Q;i++)

    {

        std::cin >> operation;

        char op = operation[0];

        switch (op) {

            case 'i':

            {

                operations[i]=op;

                cin >> value[i];

                break;

            }

            case 'c': {

                operations[i]=op;

                cin >> value[i];

                break;

            }

            case 's':

            {

                operations[i]=op;

                break;

            }

            case 'p':

            {

                operations[i]=op;

                break;

            }

            default:

                break;

            }

    }

    for(int i = 0;i<Q;i++)

    {

        switch (operations[i]) {

            case 'i': {

                bst.insert(value[i]);

                break;

            }

            case 'c': {

                std::cout << (bst.contains(value[i]) ? "Yes" : "No") << std::endl;

                break;

            }

            case 's': {

                std::cout << bst.getSize() << std::endl;

                break;

            }

            case 'p': {

                bst.print();

                break;

            }

            default:

                break;

        }

    }

    return 0;

}

*Код програми №4 Algotester Lab 78*

Завдання №4 Class Practice 1

Посилання на файл програми у пул-запиті GitHub

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/751/files#diff-565e5d0b0b9a867139f8cd959c13a3cc27c2f88352475a2bfd776314432cd4d0>

#include <iostream>

struct Node

{

    int data;

    Node \*next;

};

typedef Node\* NodePtr;

using namespace std;

NodePtr createList()

{

    return nullptr;

}

void insert(NodePtr& head,int data)

{

    NodePtr tempPtr;

    tempPtr = new Node;

    tempPtr->data = data;

    tempPtr->next = head;

    head = tempPtr;

}

NodePtr reverse(NodePtr head)

{

    NodePtr current = head;

    NodePtr prev = nullptr;

    NodePtr next = nullptr;

    while (current != nullptr)

    {

        next = current->next;

        current->next = prev;

        prev = current;

        current = next;

    }

    return prev;

}

void printList(NodePtr head)

{

    NodePtr current = head;

    if(current==nullptr)

    {

        cout<<"List is empty";

    }

    while (current != nullptr) {

        cout << current->data << " ";

        current = current->next;

    }

    cout << endl;

}

int main()

{

    NodePtr myList = createList();

    insert(myList,2);

    insert(myList,4);

    insert(myList,9);

    std::cout << "Original List: ";

    printList(myList);

    NodePtr reversedList = reverse(myList);

    std::cout << "Reversed List: ";

    printList(reversedList);

    return 0;

}

*Код програми №4 Class Practice 1*

Завдання №5 Class Practice 2

Посилання на файл програми у пул-запиті GitHub

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/751/files#diff-90d5266286208ad81d0ba97f9162db770b3b628ed2d537d170760eeb8894e8d1>

#include <iostream>

struct Node {

    int data;

    Node\* next;

};

typedef Node\* NodePtr;

void insert(NodePtr& head,int data)

{

    NodePtr tempPtr;

    tempPtr = new Node;

    tempPtr->data = data;

    tempPtr->next = head;

    head = tempPtr;

}

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

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

        if (h1->data != h2->data) {

            return false;

        }

        h1 = h1->next;

        h2 = h2->next;

    }

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

}

int main() {

    Node\* list1 = nullptr;

    insert(list1,20);

    insert(list1,11);

    insert(list1,99);

    insert(list1,39);

    insert(list1,1);

    Node\* list2 = nullptr;

    insert(list2,20);

    insert(list2,11);

    insert(list2,99);

    insert(list2,39);

    bool result = compare(list1, list2);

    if (result) {

        std::cout << "Lists are equal\n";

    } else {

        std::cout << "Lists are not equal\n";

    }

    return 0;

}

*Код програми №5 Class Practice 2*

Завдання №6 Class Practice 3

Посилання на файл програми у пул-запиті GitHub

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/751/files#diff-1787676a7bcd46a40b8f682d837e0257c6bed1254b89a8c532efec1c092fb8d3>

#include <iostream>

struct Node {

    int data;

    Node\* next;

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

};

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

    Node\* newNode = new Node(value);

    newNode->next = head;

    head = newNode;

}

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

    Node\* result = nullptr;

    Node\* current = nullptr;

    int carry = 0;

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

        int sum = carry;

        if (n1 != nullptr) {

            sum += n1->data;

            n1 = n1->next;

        }

        if (n2 != nullptr) {

            sum += n2->data;

            n2 = n2->next;

        }

        int digit = sum % 10;

        carry = sum / 10;

        if (result == nullptr) {

            result = new Node(digit);

            current = result;

        } else {

            current->next = new Node(digit);

            current = current->next;

        }

    }

    return result;

}

void printList(Node\* head) {

    while (head != nullptr) {

        std::cout << head->data << " ";

        head = head->next;

    }

    std::cout << std::endl;

}

int main() {

    Node\* num1 = nullptr;

    Node\* num2 = nullptr;

    insert(num1, 9);

    insert(num1, 7);

    insert(num1, 3);

    insert(num2, 8);

    insert(num2, 2);

    insert(num2, 4);

    Node\* result = add(num1, num2);

    std::cout << "Number 1: ";

    printList(num1);

    std::cout << "Number 2: ";

    printList(num2);

    std::cout << "Sum: ";

    printList(result);

    return 0;

}

*Код програми №6 Class Practice 3*

Завдання №7 Class Practice 4

Посилання на файл програми у пул-запиті GitHub

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/751/files#diff-d173914a604e801f018891ca27187dff04eb0c3940c3b161814ffab8bfef6257>

#include <iostream>

struct TreeNode {

    int data;

    TreeNode\* left;

    TreeNode\* right;

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

};

void insert(TreeNode\*& root, int value) {

    if (root == nullptr) {

        root = new TreeNode(value);

        return;

    }

    if (value < root->data) {

        insert(root->left, value);

    } else {

        insert(root->right, value);

    }

}

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

    if (root == nullptr) {

        return nullptr;

    }

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

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

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

    return mirroredRoot;

}

void in\_order\_traversal(TreeNode\* root) {

    if (root != nullptr) {

        in\_order\_traversal(root->left);

        std::cout << root->data << " ";

        in\_order\_traversal(root->right);

    }

}

int main() {

    TreeNode\* root = nullptr;

    insert(root, 5);

    insert(root, 3);

    insert(root, 7);

    insert(root, 2);

    insert(root, 4);

    insert(root, 6);

    insert(root, 8);

    std::cout << "Original Tree : ";

    in\_order\_traversal(root);

    std::cout << std::endl;

    TreeNode\* mirroredRoot = create\_mirror\_flip(root);

    std::cout << "Mirrored Tree : ";

    in\_order\_traversal(mirroredRoot);

    std::cout << std::endl;

    return 0;

}

*Код програми №7 Class Practice 4*

Завдання №8 Class Practice 5

Посилання на файл програми у пул-запиті GitHub

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/751/files#diff-2bc0156a915a827d4af98525f27b1636fba5d96f2533cdac75ca307ec8af2448>

#include <iostream>

struct TreeNode {

    int data;

    TreeNode\* left;

    TreeNode\* right;

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

};

void insert(TreeNode\*& root, int value) {

    if (root == nullptr) {

        root = new TreeNode(value);

        return;

    }

    if (value < root->data) {

        insert(root->left, value);

    } else {

        insert(root->right, value);

    }

}

void in\_order\_traversal(TreeNode\* root) {

    if (root != nullptr) {

        in\_order\_traversal(root->left);

        std::cout << root->data << " ";

        in\_order\_traversal(root->right);

    }

}

void tree\_sum(TreeNode\* root) {

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

        return;

    }

    tree\_sum(root->left);

    tree\_sum(root->right);

    root->data = (root->left ? root->left->data : 0) + (root->right ? root->right->data : 0);

}

int main() {

    TreeNode\* root = nullptr;

    insert(root, 5);

    insert(root, 3);

    insert(root, 7);

    insert(root, 2);

    insert(root, 4);

    insert(root, 6);

    insert(root, 8);

    std::cout << "Original Tree : ";

    in\_order\_traversal(root);

    std::cout << std::endl;

    tree\_sum(root);

    std::cout << "Tree After Sum Calculation : ";

    in\_order\_traversal(root);

    std::cout << std::endl;

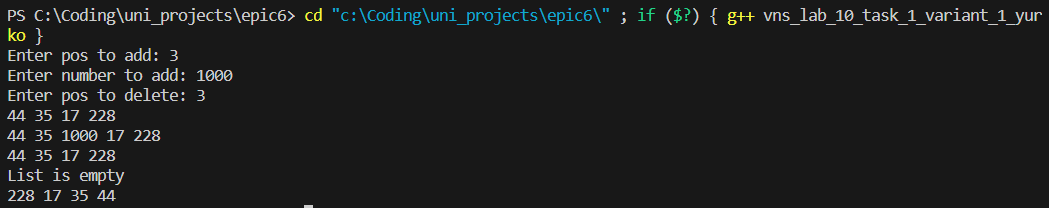
    return 0;

}

*Код програми №8 Class Practice 5*

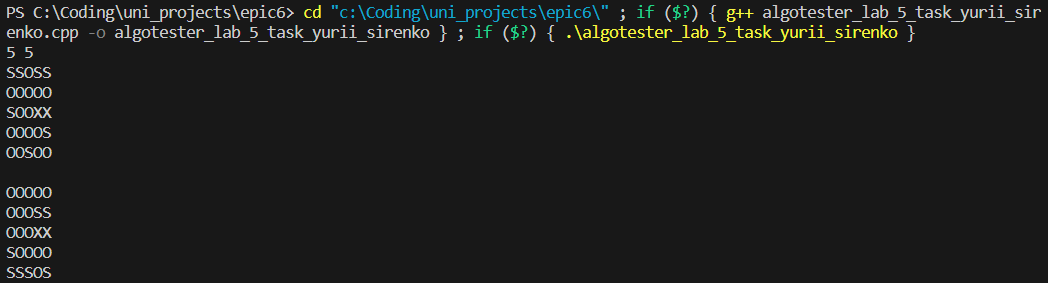
## **4. Результати виконання завдань, тестування та фактично затрачений час:**

Завдання №1 VNS Lab 10



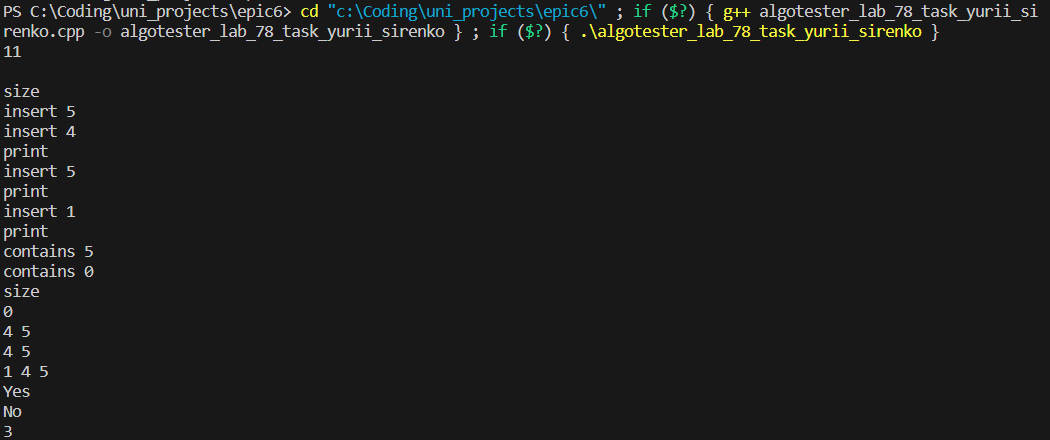
1 test VNS Lab 10

Завдання №2 Algotester Lab 5



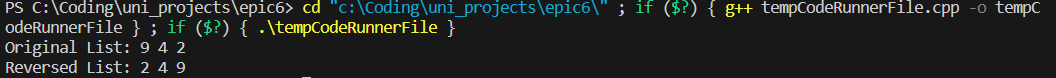
1 test Algotester Lab 5

Завдання №3 Algotester Lab 78



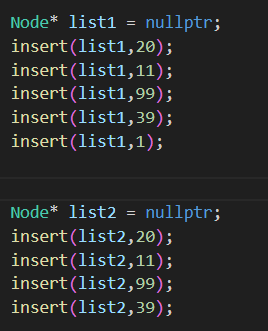
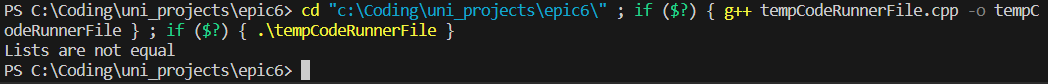
Algotester Lab 78

Завдання №4 Class Practice 1

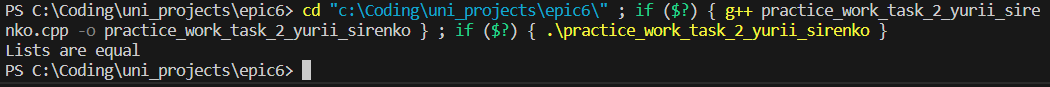
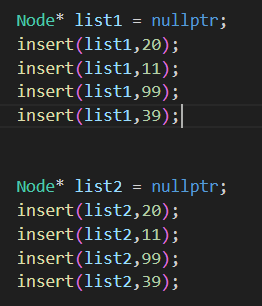


Class Practice 1

Завдання №5 Class Practice 2

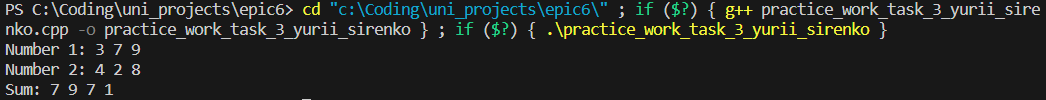
 

1 test Class Practice 2



2 test Class Practice 2

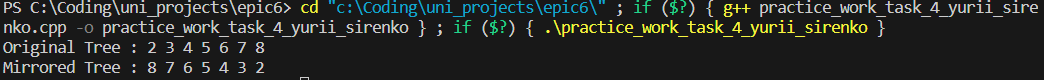
Завдання №6 Class Practice 3



1 test Class Practice 3

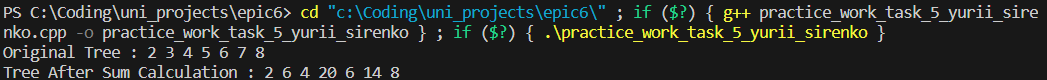
Підпис та № до блоку з виконанням та тестуванням програми

Завдання №7 Class Practice 4



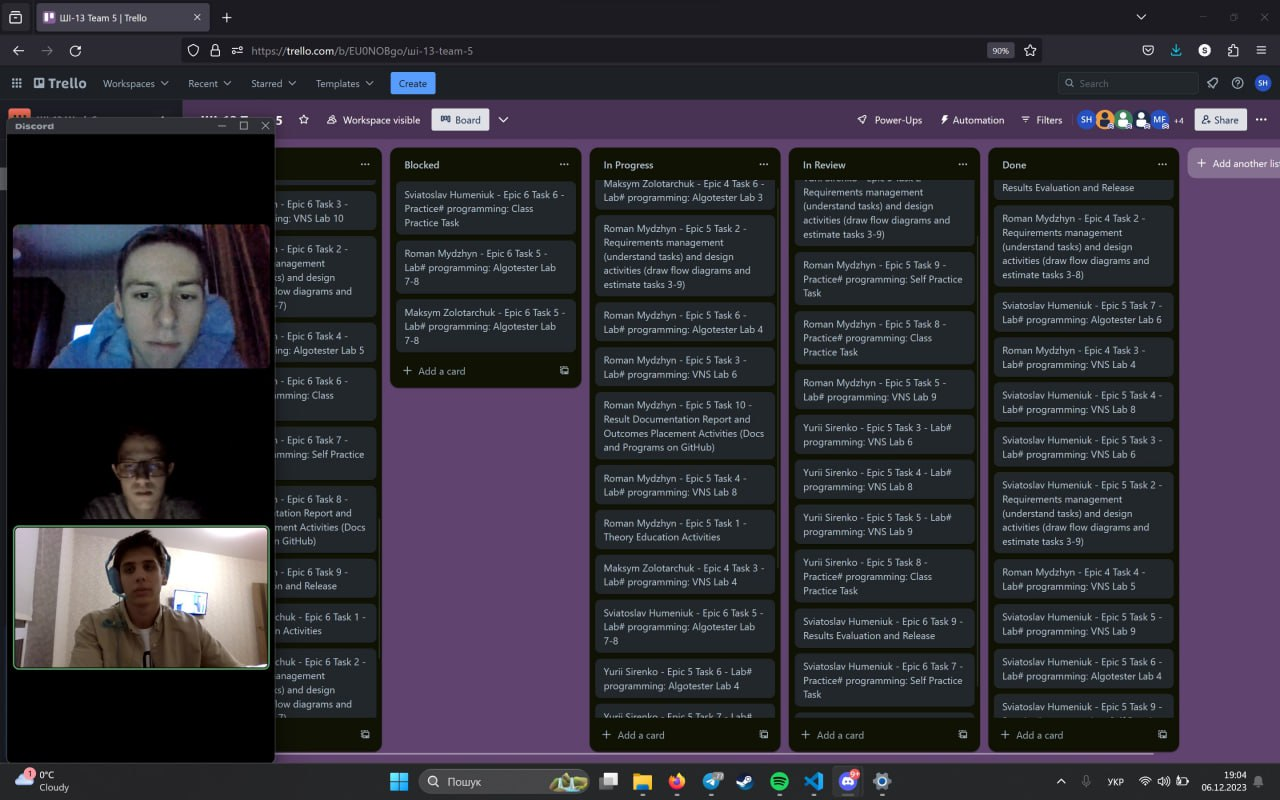
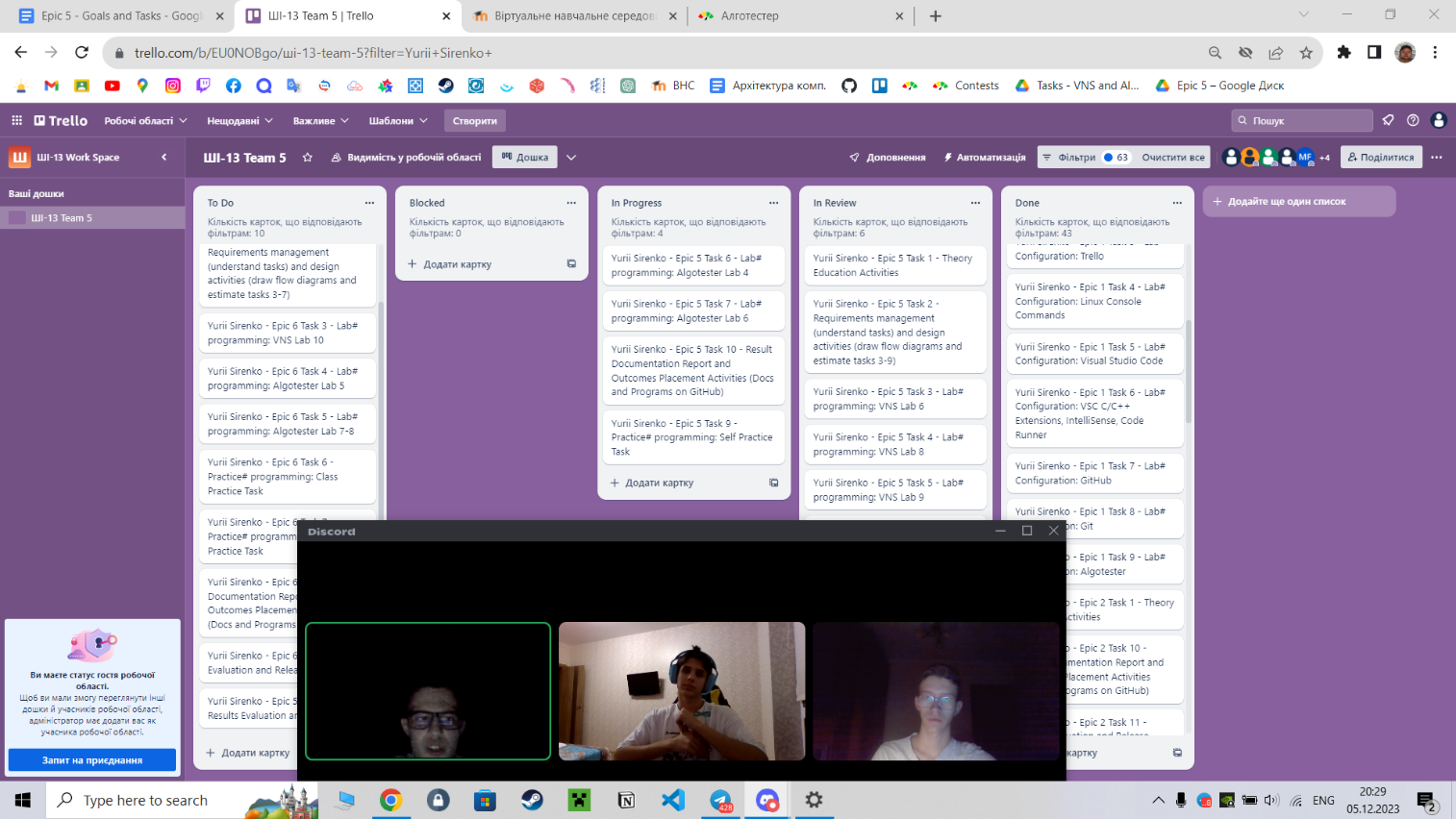
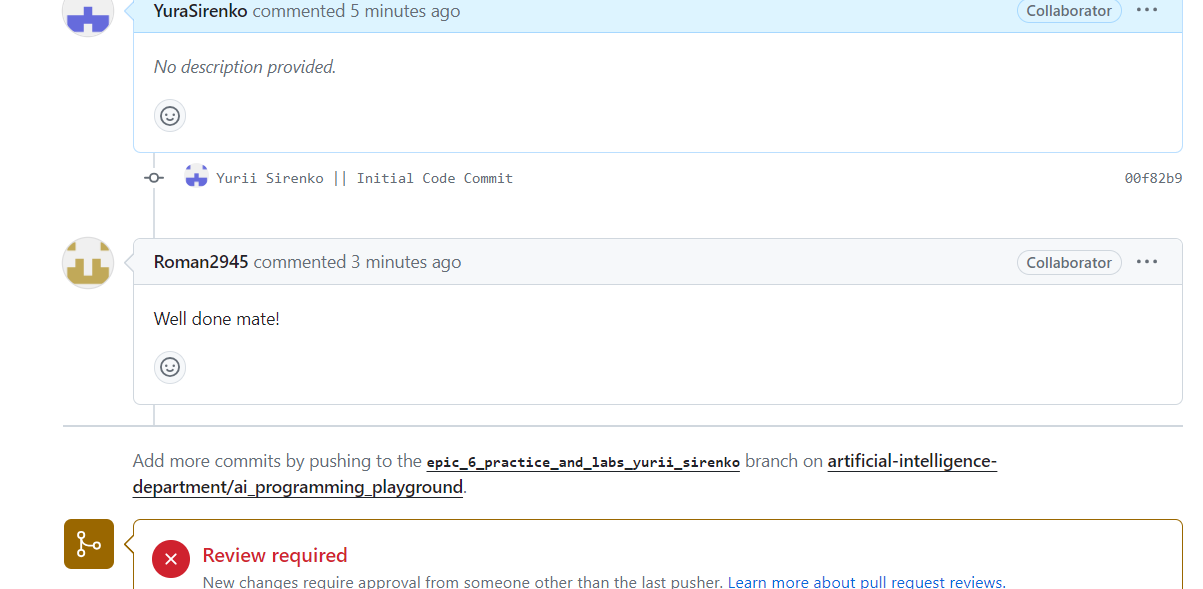
1 test Class Practice 4

Завдання №8 Class Practice 5



1 test Class Practice 5

## **5. Кооперація з командою:**

* Скрін з 1-ї зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло
* 
* Скрін з 2-ї зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло
* 
* Скрін з 2-му коментарями від учасників команди на пул реквесті з Ревю Роботи
* 

# **Висновки:**

Деталі по результатам виконання робіт та висновки згідно тем та завдань