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-13

Вітковська Марія Володимирівна

# **Тема роботи:**

Динамічні структури, такі як: черга, стек, списки(однозв’язні та двозв’язні), бінарні дерева

# **Мета роботи:**

Знайомство з динамічними структурами даних. Робота з однозв’язними та двозв’язними списками,стеками, бінарними деревами. Дослідження їх властивостей, оптимізації та застосування для ефективного зберігання та обробки динамічної інформації.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Черга
* Тема №2: Однозв’язний список
* Тема №3: Двозв’язний список
* Тема №4: Стек
* Тема №5: Бінарне дерево

1. Індивідуальний план опрацювання теорії:

**Тема №1: Черга**

* + Джерела Інформації:
    - Конспект лекцій з дисципліни «Алгоритмізація та програмування. Частина1» Л.І. Мочурад.
    - <https://www.youtube.com/playlist?list=PLVlQHNRLflP8_DGKcMoRw-TYJJALgGu4J>
    - <https://www.geeksforgeeks.org/>
    - <https://chat.openai.com/>
  + Що опрацьовано:
    - Додавання елементів в чергу та видалення з неї, отримання розміру та очищення черги, отримання доступу до елемента на передньому кінці черги. Приклади використання черги та використання її для виконання поставлених задач. Різниця між чергою та стеком.
  + Статус: Ознайомлена
  + Початок опрацювання теми: 12.12
  + Завершення опрацювання теми: 12.12

**Тема №2: Однозв’язний список**

* + Джерела Інформації:
    - Конспект лекцій з дисципліни «Алгоритмізація та програмування. Частина1» Л.І. Мочурад.
    - <https://www.youtube.com/playlist?list=PLVlQHNRLflP8_DGKcMoRw-TYJJALgGu4J>
    - <https://www.geeksforgeeks.org/>
    - <https://chat.openai.com/>
    - <https://www.youtube.com/watch?v=-25REjF_atI>
  + Що опрацьовано:
    - Визначення однозв’язного списку, ключові аспекти, такі як: структура вузла списку, додавання елементів в кінець списку, вивід елементів і тд. Методи списку. Приклади використання однозв’язних списків
  + Статус: Ознайомлена
  + Початок опрацювання теми: 12.12
  + Завершення опрацювання теми: 12.12

**Тема №3: Двозв’язний список**

* + Джерела Інформації:
    - Конспект лекцій з дисципліни «Алгоритмізація та програмування. Частина1» Л.І. Мочурад.
    - <https://www.geeksforgeeks.org/>
    - <https://chat.openai.com/>
    - <https://www.youtube.com/watch?v=QLzu2-_QFoE&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=141>
  + Що опрацьовано:
    - Визначення однозв’язного списку, ключові аспекти, такі як: структура вузла списку – початок(head) і кінець (tail), методи списку.
  + Статус: Ознайомлена
  + Початок опрацювання теми: 13.12
  + Завершення опрацювання теми: 13.12

**Тема №4: Стек**

* + Джерела Інформації:
    - Конспект лекцій з дисципліни «Алгоритмізація та програмування. Частина1» Л.І. Мочурад.
    - <https://www.youtube.com/playlist?list=PLVlQHNRLflP8_DGKcMoRw-TYJJALgGu4J>
    - <https://chat.openai.com/>
    - <https://www.youtube.com/watch?v=ZYvYISxaNL0&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=143>
    - <https://www.youtube.com/watch?v=QLzu2-_QFoE&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=142>
  + Що опрацьовано:
    - Додавання елементів в стек та видалення з нього, отримання розміру та очищення стеку, отримання доступу до елемента на передньому кінці черги. Приклади використання стеків та використання для виконання поставлених задач. Різниця між стеком та чергою.
* Статус: Ознайомлена
  + Початок опрацювання теми: 13.12
  + Завершення опрацювання теми: 13.12
* **Тема №5: Бінарне дерево**
  + Джерела Інформації:
    - Конспект лекцій з дисципліни «Алгоритмізація та програмування. Частина1» Л.І. Мочурад.
    - <https://www.youtube.com/playlist?list=PLVlQHNRLflP8_DGKcMoRw-TYJJALgGu4J>
    - <https://chat.openai.com/>
    - <https://www.youtube.com/watch?v=qBFzNW0ALxQ>
  + Що опрацьовано:
    - Визначення бінарного дерева, бінарне дерево пошуку(BST), обхід дерев, пошук в довжину(DFS) та ширину(BFS) , видалення вузла з дерева. AVL- дерево та червоно-чорне дерево. Збалансоване дерево.
  + Статус: Ознайомлена
  + Початок опрацювання теми: 14.12
  + Завершення опрацювання теми: 15.12

# **Виконання роботи:**

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

**Завдання №1 Опрацювання теорії**

* Деталі завдання: опрацювання різноманітних матеріалів, аналіз та дослідження відео, статей, книг на задані теми. Практика з простими прикладами та задачками для освоєння отриманого матеріалу Організація та структурування отриманих даних для можливості ефективніше засвоювати отримані знання на практиці. Вивчення найважливіших моментів.

**Завдання №2 Вимоги та проектування за допомогою Draw.io(складання блок-схем)**

* Деталі завдання: введення завдання в Trello та їхнє переміщення в відповідну колонку в процесі виконання роботи, побудова блок-схем до завдань.

**Завдання №3 Лабораторна №10 з VNS**

* Варіант завдання: **24**
* Деталі завдання:

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.

Записи в лінійному списку містять ключове поле типу \*char (рядок символів). Сформувати двонаправлений список. Знищити елемент із заданим номером. Додати по К елементів перед і після елемента із заданим ключем.

**Завдання №4 Algotester Lab 5, варіант 3**

* Деталі завдання: У вас є карта гори розміром N×M.

Також ви знаєте координати {x,y}, у яких знаходиться вершина гори.

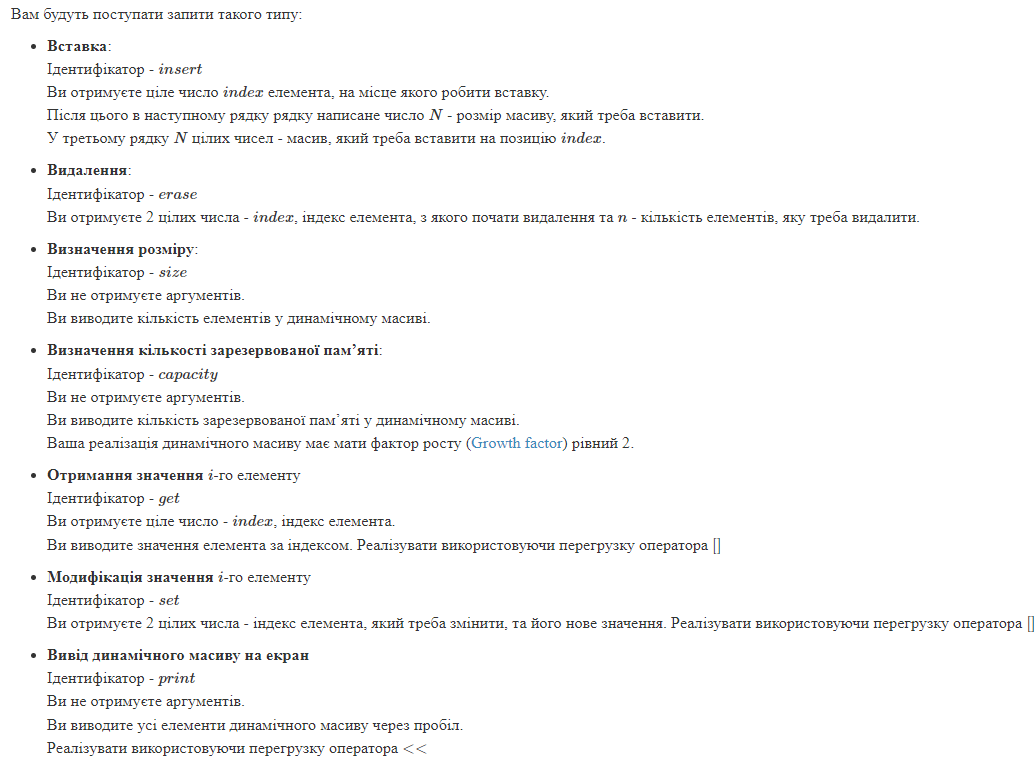
Ваше завдання - розмалювати карту таким чином, щоб найнижча точка мала число 0, а пік гори мав найбільше число.

Клітинки, які мають суміжну сторону з вершиною мають висоту на один меншу, суміжні з ними і не розфарбовані мають ще на 1 меншу висоту і так далі.

**Завдання №7 Algotester Lab 7, варіант 2**

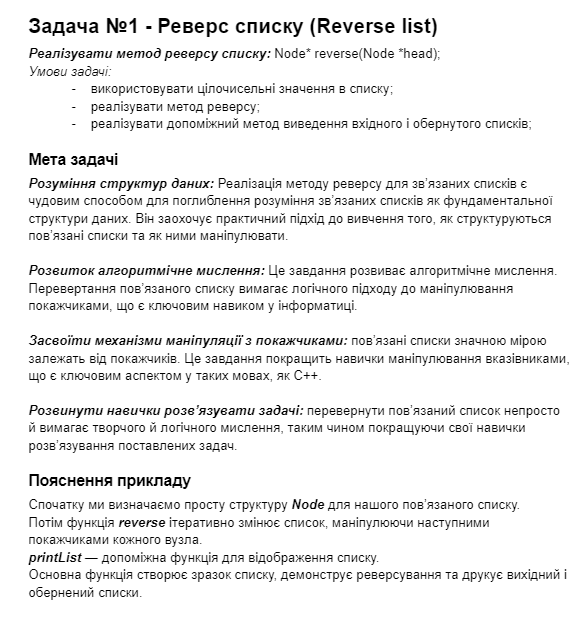
* Деталі завдання: Ваше завдання - власноруч реалізувати структуру даних "Динамічний

масив". Ви отримаєте Q запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його аргументи.



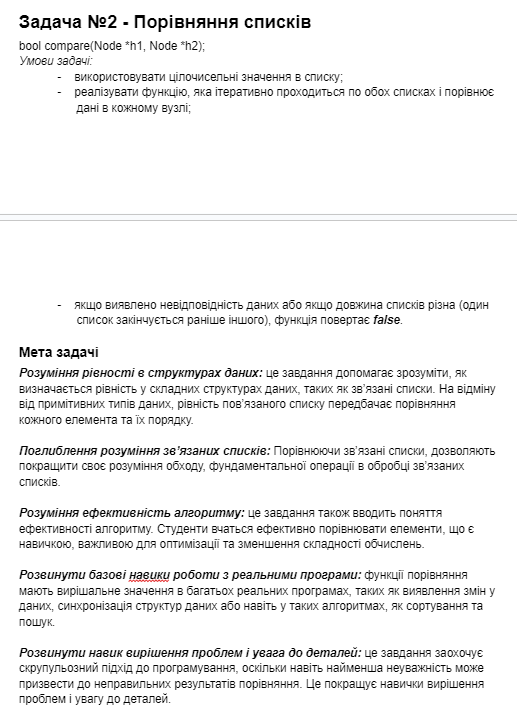
Малюнок Algo 7v8\_2 condition

**Завдання №8 Class Practice Task 1**

* Деталі завдання: 

Малюнок Class Practice Task 1 condition

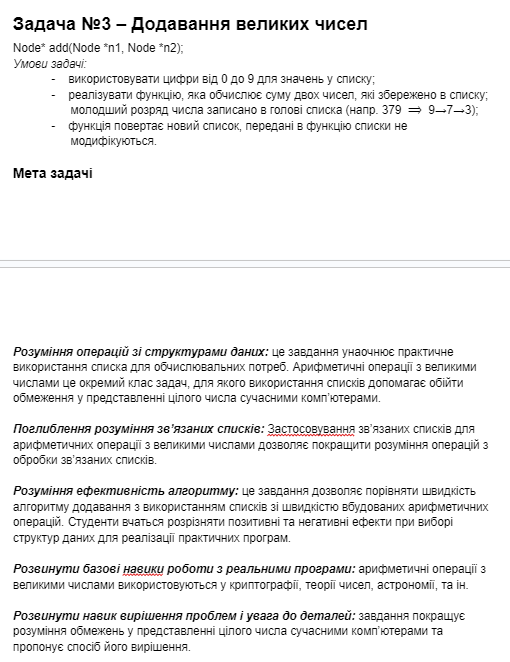
**Завдання №9 Class Practice Task 2**

* Деталі завдання:
* 

Малюнок Class Practice Task 2 condition

**Завдання №10 Class Practice Task 3**

* Деталі завдання:

******

Малюнок Class Practice Task 3 condition

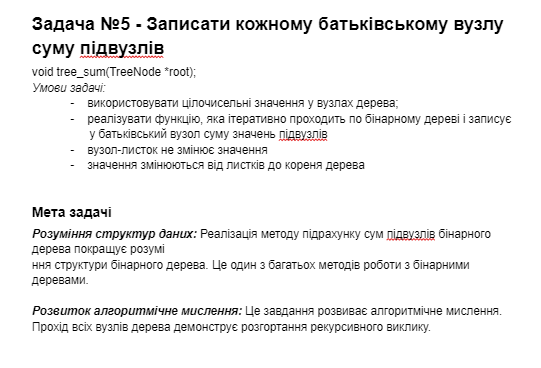
**Завдання №11 Class Practice Task 4**

* Деталі завдання: 

Малюнок Class Practice Task 4 condition

**Завдання №12 Class Practice Task 5**

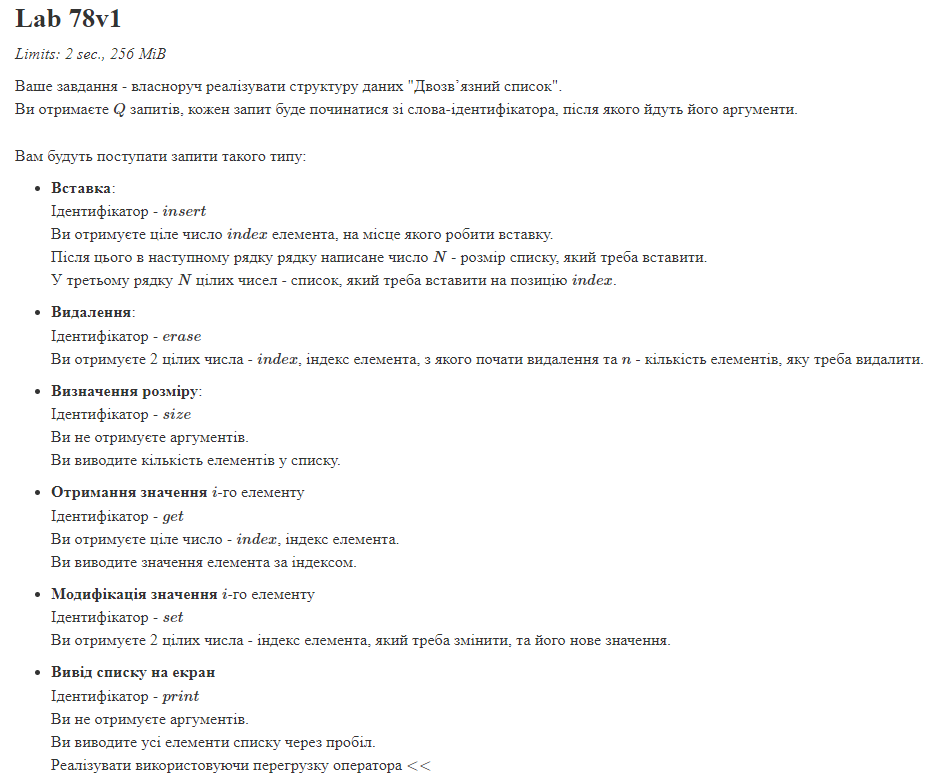
* Деталі завдання:

******

Малюнок Class Practice Task 5 condition

**Завдання №13 Self-Practice Task**

* Деталі завдання:

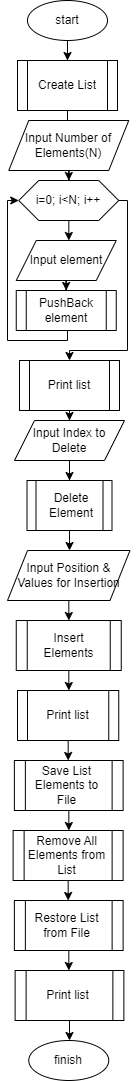
******

Малюнок Self Practice Task condition

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

Програма №1 **VNS Lab\_10**

* Блок-схема:



Малюнок Блок-схема VNS lab 10

**3. Код програм з посиланням на зовнішні ресурси:**

**Завдання №1 Лабораторна №10 з VNS, варіант 24**

#include <iostream>

#include <fstream>

#include <string>

#include <cstring>

using namespace std;

struct ListNode {

    char data[100];

    ListNode\* prev;

    ListNode\* next;

};

ListNode\* createList() {

    return nullptr;

}

void appendNode(ListNode\*& head, const char\* data) {

    ListNode\* newNode = new ListNode;

    strcpy(newNode->data, data);

    newNode->next = nullptr;

    if (head == nullptr) {

        newNode->prev = nullptr;

        head = newNode;

    } else {

        ListNode\* temp = head;

        while (temp->next != nullptr) {

            temp = temp->next;

        }

        temp->next = newNode;

        newNode->prev = temp;

    }

}

void deleteByIndex(ListNode\*& head, int index) {

    if (head == nullptr) {

        cout << "The list is empty\n";

        return;

    }

    ListNode\* temp = head;

    for (int i = 0; i < index && temp != nullptr; i++) {

        temp = temp->next;

    }

    if (temp == nullptr) {

        cout << "Node with index " << index << " does not exist\n";

        return;

    }

    if (temp->prev != nullptr) {

        temp->prev->next = temp->next;

    } else {

        head = temp->next;

    }

    if (temp->next != nullptr) {

        temp->next->prev = temp->prev;

    }

    delete temp;

}

void printList(ListNode\* head) {

    if (head == nullptr) {

        cout << "The list is empty\n";

        return;

    }

    ListNode\* temp = head;

    while (temp != nullptr) {

        cout << temp->data << " ";

        temp = temp->next;

    }

    cout << endl;

}

void writeListToFile(ListNode\* head, const char\* filename) {

    ofstream outFile(filename);

    if (!outFile.is\_open()) {

        cout << "Error opening the file\n";

        return;

    }

    ListNode\* temp = head;

    while (temp != nullptr) {

        outFile << temp->data << endl;

        temp = temp->next;

    }

    outFile.close();

}

void deleteList(ListNode\*& head) {

    while (head != nullptr) {

        ListNode\* temp = head;

        head = head->next;

        delete temp;

    }

}

ListNode\* restoreFromFile(const char\* filename) {

    ListNode\* head = nullptr;

    ifstream inFile(filename);

    if (!inFile.is\_open()) {

        cout << "Error opening the file\n";

        return nullptr;

    }

    char data[100];

    while (inFile.getline(data, sizeof(data))) {

        appendNode(head, data);

    }

    inFile.close();

    return head;

}

int main() {

    ListNode\* myList = createList();

    appendNode(myList, "Data1");

    appendNode(myList, "Data2");

    appendNode(myList, "Data3");

    cout << "List before changes:\n";

    printList(myList);

    deleteByIndex(myList, 1);

    cout << "List after deletion:\n";

    printList(myList);

    appendNode(myList, "Data4");

    cout << "List after adding a node:\n";

    printList(myList);

    writeListToFile(myList, "myList.txt");

    deleteList(myList);

    cout << "List after deletion (should be empty):\n";

    printList(myList);

    myList = restoreFromFile("myList.txt");

    cout << "Restored list:\n";

    printList(myList);

    deleteList(myList);

    return 0;

}

VNS Lab\_10\_V24

**Завдання №2 Algotester Lab 5, варіант 3**

#include <iostream>

#include <vector>

#include <queue>

using namespace std;

void paint\_the\_landscape(int N, int M, int X, int Y) {

    vector<vector<int>> map(N, vector<int>(M, -1));

    queue<pair<int, int>> queue\_mountain;

    queue\_mountain.push({X - 1, Y - 1});

    map[X - 1][Y - 1] = 0;

    const int x[] = {1, -1, 0, 0};

    const int y[] = {0, 0, 1, -1};

    while (!queue\_mountain.empty()) {

        pair<int, int> current = queue\_mountain.front();

        queue\_mountain.pop();

        for (int i = 0; i < 4; ++i) {

            int nx = current.first + x[i];

            int ny = current.second + y[i];

            if (nx >= 0 && nx < N && ny >= 0 && ny < M && map[nx][ny] == -1) {

                map[nx][ny] = map[current.first][current.second] + 1;

                queue\_mountain.push({nx, ny});

            }

        }

    }

    int max\_height = 0;

    for (int i = 0; i < N; ++i) {

        for (int j = 0; j < M; ++j) {

            max\_height = max(max\_height, map[i][j]);

        }

    }

    for (int i = 0; i < N; ++i) {

        for (int j = 0; j < M; ++j) {

            cout << max\_height - map[i][j] << " ";

        }

        cout << endl;

    }

}

int main() {

    int Min = 1;

    int Max = 1000;

    int N, M, X, Y;

    cin >> N >> M;

    cin >> X >> Y;

    if (N >= Min && N <= Max && M >= Min && M <= Max &&

        X >= Min && X <= N && Y >= Min && Y <= M) {

        paint\_the\_landscape(N, M, X, Y);

    }

    else {

        cout << "Invalid input, please try again.";

    }

    return 0;

}

VNS Lab8 Task1\_V24

**Завдання №3 Algotester Lab 7 варіант 2**

#include <iostream>

#include <string>

using namespace std;

template<typename Type>

struct DynamicArray {

    int size = 0;

    int capacity = 1;

    Type \*A;

    DynamicArray() {

        A = new Type[1];

    }

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

        if (capacity > size + N) {

            for (int i = size - 1; i > index - 1; i--) {

                A[i + N] = A[i];

            }

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

                A[i+index]=arr[i];

            }

            size += N;

        } else {

            while (capacity <= size + N) {

                capacity \*= 2;

            }

            Type \*B = new Type[capacity];

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

                B[i] = A[i];

            }

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

                B[i+index]=arr[i];

            }

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

                B[i + N] = A[i];

            }

            delete[] A;

            A = B;

            size += N;

        }

    }

    void Erase(int index, int n) {

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

            A[i] = A[i + n];

        }

        size -= n;

    }

    int Size() {

        return size;

    }

    int Capacity() {

        return capacity;

    }

    Type &operator[](int index) {

        return A[index];

    }

    friend ostream &operator<<(ostream &os, const DynamicArray &array) {

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

            os << array.A[i] << ' ';

        }

        return os;

    }

};

int main() {

    DynamicArray<int> A;

    int Q;

    cin >> Q;

    string ident;

    for (int i = 0; i < Q; ++i) {

        cin >> ident;

        if (ident == "insert") {

            int index, N;

            cin >> index >> N;

            int arr[N];

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

                cin >> arr[i];

            }

            A.Insert(index, arr, N);

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

            int index, n;

            cin >> index >> n;

            A.Erase(index, n);

        } else if (ident == "size") {

            cout << A.Size() << endl;

        } else if (ident == "capacity") {

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

        } else if (ident == "get") {

            int index;

            cin >> index;

            cout << A[index] << endl;

        } else if (ident == "set") {

            int index;

            cin >> index;

            cin >> A[index];

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

            cout << A << endl;

        }

    }

    return 0;

}

Algo 7v8\_2

**Завдання №4 Class Practice Task №6 1**

#include <iostream>

using namespace std;

struct Node {

    int data;

    Node\* next;

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

};

// Метод для реверсу списку

Node\* reverse(Node\* head) {

    Node \*prev = nullptr, \*current = head, \*nextNode = nullptr;

    while (current != nullptr) {

        nextNode = current->next;

        current->next = prev;

        prev = current;

        current = nextNode;

    }

    return prev;

}

// Допоміжна функція для виведення списку

void printList(Node\* head) {

    while (head != nullptr) {

        cout << head->data << " ";

        head = head->next;

    }

    cout << endl;

}

int main() {

    int n;

    cout << "Enter the number of nodes: ";

    cin >> n;

    if (n <= 0) {

        cout << "Invalid input. Number of nodes should be greater than 0." << endl;

        return 1; // Вихід з програми з помилкою

    }

    Node\* head = nullptr;

    Node\* current = nullptr;

    // Зчитуємо значення для кожного вузла

    for (int i = 0; i < n; ++i) {

        int value;

        cout << "Enter value for node " << i + 1 << ": ";

        cin >> value;

        if (i == 0) {

            head = new Node(value);

            current = head;

        } else {

            current->next = new Node(value);

            current = current->next;

        }

    }

    // Виведення вихідного списку

    cout << "Original List: ";

    printList(head);

    // Реверс списку

    head = reverse(head);

    // Виведення оберненого списку

    cout << "Reversed List: ";

    printList(head);

    return 0;

}

Class Practice Task1

**Завдання №5 Class Practice Task №6 2**

#include <iostream>

struct Node {

    int data;

    Node\* next;

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

};

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

    while (h1 && h2) {

        if (h1->data != h2->data) {

            return false;

        }

        h1 = h1->next;

        h2 = h2->next;

    }

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

}

// Функція для створення списку заданого розміру

Node\* createList(int size) {

    if (size <= 0) {

        return nullptr;

    }

    Node\* head = nullptr;

    Node\* current = nullptr;

    for (int i = 0; i < size; ++i) {

        int value;

        std::cout << "Enter the value of the node " << i + 1 << ": ";

        std::cin >> value;

        if (head == nullptr) {

            head = new Node(value);

            current = head;

        } else {

            current->next = new Node(value);

            current = current->next;

        }

    }

    return head;

}

int main() {

    int size1, size2;

    std::cout << "Enter the number of nodes for the first list: ";

    std::cin >> size1;

    Node\* head1 = createList(size1);

    std::cout << "Enter the number of nodes for the second list: ";

    std::cin >> size2;

    Node\* head2 = createList(size2);

    bool result = compare(head1, head2);

    if (result) {

        std::cout << "The lists are identical.\n";

    } else {

        std::cout << "The lists differ.\n";

    }

    // Звільнення пам'яті

    while (head1 != nullptr) {

        Node\* temp = head1;

        head1 = head1->next;

        delete temp;

    }

    while (head2 != nullptr) {

        Node\* temp = head2;

        head2 = head2->next;

        delete temp;

    }

    return 0;

}

Class Practice Task2

**Завдання №6 Class Practice Task №6 3**

#include <iostream>

#include <cmath>

using namespace std;

struct Node {

    int data;

    Node\* next;

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

};

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

    Node\* newNode = new Node(value);

    newNode->next = head;

    head = newNode;

}

int convert(Node\* h, int N) {

    int result = 0;

    Node\* current = h;

    for (int i = 0; i < N; i++) {

        result = result \* 10 + current->data;

        current = current->next;

    }

    return result;

}

int main() {

    int N, n, result = 0;

    Node\* head1 = nullptr;

    Node\* head2 = nullptr;

    cout << "How many numbers do you want to add in the first list? ";

    cin >> N;

    cout << "Enter the first list: ";

    for (int i = 0; i < N; i++) {

        int k;

        cin >> k;

        push(head1, k);

    }

    cout << "How many numbers do you want to add in the second list? ";

    cin >> n;

    cout << "Enter the second list: ";

    for (int i = 0; i < n; i++) {

        int k;

        cin >> k;

        push(head2, k);

    }

    result = convert(head1, N) + convert(head2, n);

    cout << "Sum of the two lists: " << result << endl;

    return 0;

}

Class Practice Task3

**Завдання №7 Class Practice Task №6 4**

#include <iostream>

#include <queue>

using namespace std;

struct TreeNode {

    int data;

    TreeNode \*left;

    TreeNode \*right;

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

};

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

    if (root == nullptr) {

        return nullptr;

    }

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

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

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

    return newRoot;

}

void levelOrderTraversal(TreeNode \*root) {

    if (root == nullptr) {

        return;

    }

    queue<TreeNode \*> q;

    q.push(root);

    while (!q.empty()) {

        TreeNode \*current = q.front();

        q.pop();

        cout << current->data << " ";

        if (current->left) {

            q.push(current->left);

        }

        if (current->right) {

            q.push(current->right);

        }

    }

}

int main() {

    TreeNode \*root = new TreeNode(1);

    root->left = new TreeNode(2);

    root->right = new TreeNode(3);

    root->left->left = new TreeNode(4);

    root->left->right = new TreeNode(5);

    cout << "Original tree: ";

    levelOrderTraversal(root);

    cout << endl;

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

    cout << "Mirrored tree: ";

    levelOrderTraversal(mirroredRoot);

    cout << endl;

    return 0;

}

Class Practice Task4

**Завдання №8 Class Practice Task №6 5**

#include <iostream>

using namespace std;

struct TreeNode {

    int data;

    TreeNode \*left, \*right;

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

};

// Функція для обчислення суми підвузлів і збереження результату у батьківському вузлі

void tree\_sum(TreeNode \*root) {

    if (root == nullptr) {

        return;

    }

    // Рекурсивно викликаємо функцію для лівого і правого піддерев

    tree\_sum(root->left);

    tree\_sum(root->right);

    if (root->left != nullptr) {

        root->data += root->left->data;

    }

    if (root->right != nullptr) {

        root->data += root->right->data;

    }

}

// Функція для виведення значень дерева в порядку post-order

void printTree(TreeNode \*root) {

    if (root == nullptr) {

        return;

    }

    printTree(root->left);

    printTree(root->right);

    cout << root->data << " ";

}

int main() {

    TreeNode \*root = new TreeNode(1);

    root->left = new TreeNode(2);

    root->right = new TreeNode(3);

    root->left->left = new TreeNode(4);

    root->left->right = new TreeNode(5);

    root->right->left = new TreeNode(6);

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

    cout << "Tree before calculating the sum of subnodes: ";

    printTree(root);

    cout << endl;

    // Виклик функції для обчислення суми підвузлів

    tree\_sum(root);

    cout << "Tree after calculating the sum of subnodes: ";

    printTree(root);

    cout << endl;

    return 0;

}

Class Practice Task5

**Завдання №9 Self Practice Task**

#include <iostream>

using namespace std;

template <typename T>

class Node {

public:

    T data;

    Node\* next;

    Node\* prev;

    Node(T val) : data(val), next(nullptr), prev(nullptr) {}

};

template <typename T>

class DoublyLinkedList {

private:

    Node<T>\* head;

    Node<T>\* tail;

    int size;

public:

    DoublyLinkedList() : head(nullptr), tail(nullptr), size(0) {}

    ~DoublyLinkedList() {

        clear();

    }

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

        if (index < 0 || index > size) {

            return;

        }

        Node<T>\* current = getNode(index - 1);

        for (int i = 0; i < N; ++i) {

            Node<T>\* newNode = new Node<T>(values[i]);

            if (current == nullptr) {

                newNode->next = head;

                if (head != nullptr) {

                    head->prev = newNode;

                }

                head = newNode;

                if (tail == nullptr) {

                    tail = newNode;

                }

            } else {

                newNode->next = current->next;

                newNode->prev = current;

                if (current->next != nullptr) {

                    current->next->prev = newNode;

                } else {

                    tail = newNode;

                }

                current->next = newNode;

            }

            ++size;

            current = newNode;

        }

    }

    void erase(int index, int n) {

        if (index < 0 || index >= size || n <= 0) {

            return;

        }

        Node<T>\* current = getNode(index);

        for (int i = 0; i < n && current != nullptr; ++i) {

            Node<T>\* temp = current;

            current = current->next;

            if (temp->prev != nullptr) {

                temp->prev->next = temp->next;

            } else {

                head = temp->next;

                if (head != nullptr) {

                    head->prev = nullptr;

                }

            }

            if (temp->next != nullptr) {

                temp->next->prev = temp->prev;

            } else {

                tail = temp->prev;

                if (tail != nullptr) {

                    tail->next = nullptr;

                }

            }

            delete temp;

            --size;

        }

    }

    int Size() const {

        return size;

    }

    T get(int index) const {

        Node<T>\* node = getNode(index);

        if (node != nullptr) {

            return node->data;

        } else {

            return T();

        }

    }

    void set(int index, const T& value) {

        Node<T>\* node = getNode(index);

        if (node != nullptr) {

            node->data = value;

        }

    }

    void print(ostream& os) const {

        Node<T>\* current = head;

        while (current != nullptr) {

            os << current->data << ' ';

            current = current->next;

        }

    }

    void clear() {

        Node<T>\* current = head;

        while (current != nullptr) {

            Node<T>\* temp = current;

            current = current->next;

            delete temp;

        }

        head = tail = nullptr;

        size = 0;

    }

private:

    Node<T>\* getNode(int index) const {

        if (index < 0 || index >= size) {

            return nullptr;

        }

        Node<T>\* current = head;

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

            current = current->next;

        }

        return current;

    }

};

// Overload the << operator for easy printing

template <typename T>

ostream& operator<<(ostream& os, const DoublyLinkedList<T>& list) {

    list.print(os);

    return os;

}

int main() {

    DoublyLinkedList<int> list;

    int q;

    cin >> q;

    for (int i = 0; i < q; ++i) {

        string command;

        cin >> command;

        if (command == "insert") {

            int index, N;

            cin >> index >> N;

            int\* values = new int[N];

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

                cin >> values[j];

            }

            list.insert(index, values, N);

            delete[] values;

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

            int index, n;

            cin >> index >> n;

            list.erase(index, n);

        } else if (command == "size") {

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

        } else if (command == "get") {

            int index;

            cin >> index;

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

        } else if (command == "set") {

            int index, value;

            cin >> index >> value;

            list.set(index, value);

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

            list.print(cout);

            cout << endl;

        }

    }

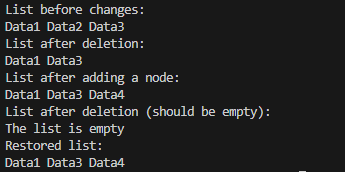
    list.clear();

    return 0;

}

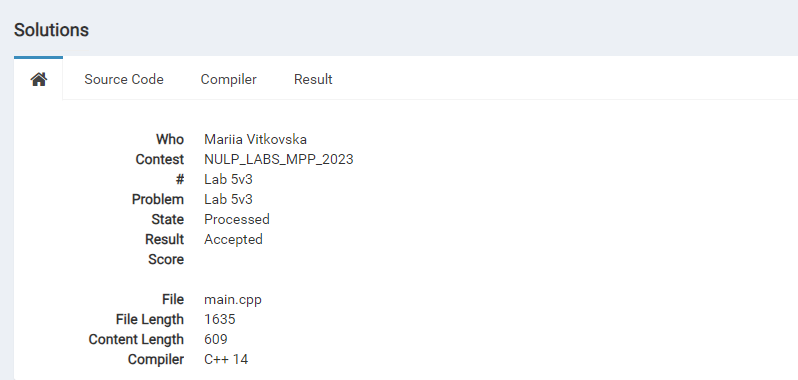
Self Practice Task(Algotester\_lab\_7v8\_1)

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

Завдання №1 **Лабораторна №10 з VNS, завдання 1 варіант 24** ****

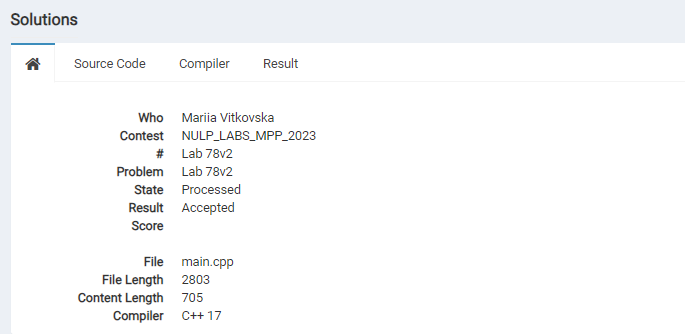
Час затрачений на виконання завдання: 4 години

Завдання №4 **Algotester Lab 5, варіант 3**



Малюнок Algo 5\_3 results

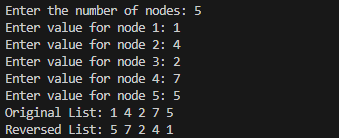
Час затрачений на виконання завдання: 9 годин

Завдання №5 **Algotester Lab 7, варіант 2**

Малюнок Algo 7\_2 result

Час затрачений на виконання завдання: 12 годин

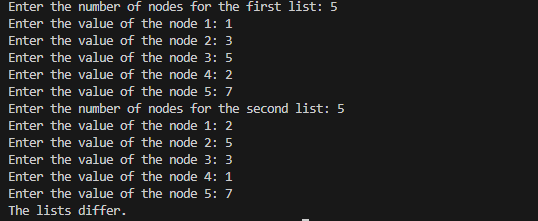
Завдання №6 **Class-Practice Task 1**



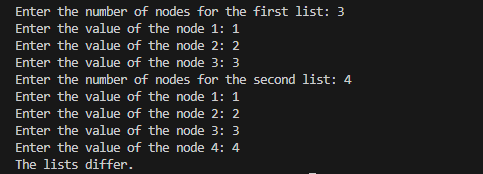
Малюнок Practice 1 result

Час затрачений на виконання завдання: 2 години

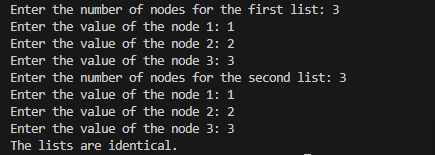
Завдання №6 **Class-Practice Task 2**



Малюнок Practice 2 result



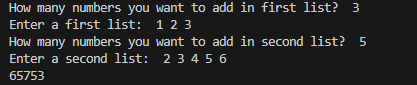
Малюнок Practice 2 result



Малюнок Practice 2 result

Час затрачений на виконання завдання: 2 години

Завдання №6 **Class-Practice Task 3**



Малюнок Practice 3 result

Час затрачений на виконання завдання: 2 години

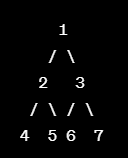
Завдання №6 **Class-Practice Task 4**



Малюнок Practice 4 result

Час затрачений на виконання завдання: 2 години

Завдання №6 **Class-Practice Task 5**



Малюнок Our tree at the beginning

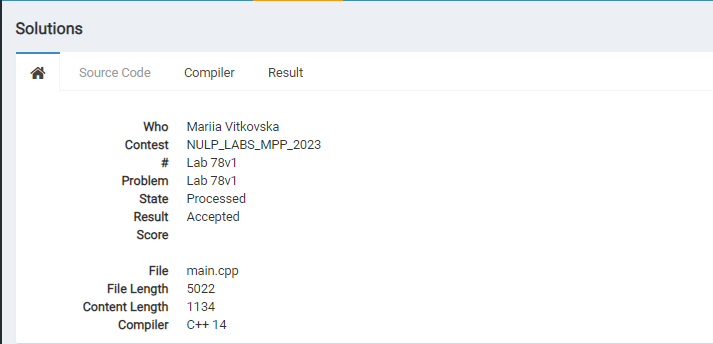


Малюнок Practice 5 result

Ми виводимо дерево перед додаванням підвузлів в такому порядку: ліві підвузли, праві підвузли, батьківський вузол.

Час затрачений на виконання завдання: 1 година

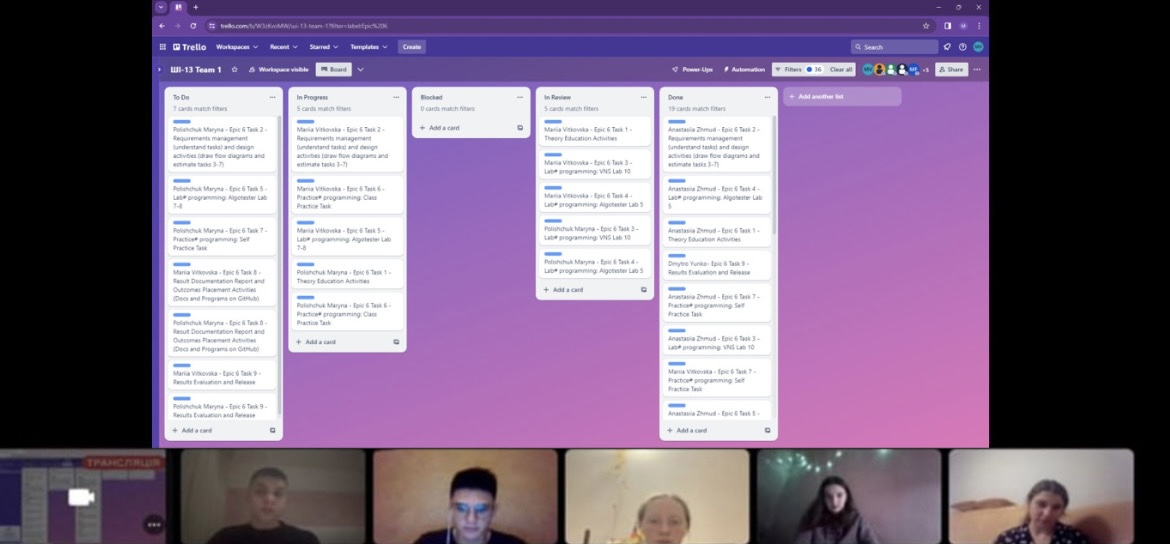
Завдання №7 **Self-Practice Task**



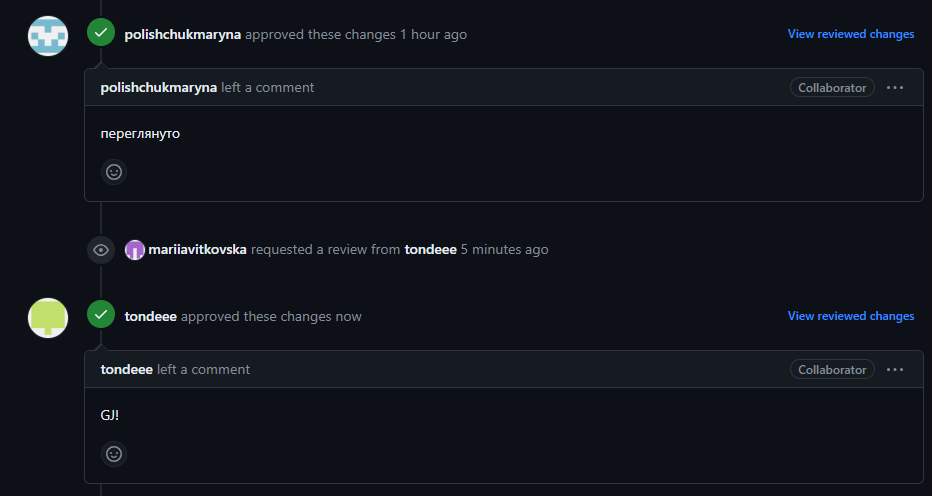
Малюнок Self Practice result

Час затрачений на виконання завдання: 3 години

## **5. Кооперація з командою:**



Малюнок Скрін зустрічі з командою при обговоренні тасків в Епіку №6 та прогресу в Trello



Малюнок Апрув від тімейтів

**Pull-request:** [**https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/1169**](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1169)

# **Висновки:**

В процесі вивчення тем «Черга», «Стек», «Однозв’язний та двозв’язний списки», «Бінарне дерево» було отримано всі теоретичні знання, які включають основи одновимірних та двовимірних масивів, роботу з вказівниками та методи обробки масивів за допомогою функцій, і виконано додаткові нескладні завдання для освоєння матеріалу для подальшого виконання практичних задач у цій лабораторній роботі. Ці навички надали змогу ефективно розв’язати поставлені задачі та покращити якість коду. Виконавши всі надані завдання, було засвоєно теоретичний матеріал, який можна буде ефективно використовувати в подальших роботах та завданнях.