Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт №10**

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

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

***Виконав:***

студент групи ШІ-14

Чумаченко Дем’ян Сергійович

# **Тема роботи:**

Динамічні структури даних та алгоритми їх обробки. Однозв’язний та двозв’язний список. Стек, черга та дерево.

# **Мета роботи:**

Ознайомитись з динамічними структурами даних та попрацювати з базовими алгоритмами їх обробки. Реалізувати однозв’язний та двозв’язний списки та створити функції для їх обробки. Ознайомитися з деревами та реалізувати бінарне дерево. Ознайомитися з стеком та чергою

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Стек. Черга.
* Тема №2: Однозв’язний та двозв’язний списки.
* Тема №3: Дерева. Бінарне дерево.
* Тема №4: Класи. Шаблони класів.
* Тема №5: Алгоритми обробки динамічних структур даних.

1. Індивідуальний план опрацювання теорії:

Тема №1: Однозв’язний та двозв’язний списки.

* + Джерела Інформації:
    - Відео.

<https://www.youtube.com/watch?v=-25REjF_atI&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>

<https://www.youtube.com/watch?v=QLzu2-_QFoE&t=1328s&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>

* + Що опрацьовано:
    - Зв’язані списки. Різниця між однозв’язним та двозв’язними списками
  + Статус: Ознайомлений
  + Початок опрацювання теми: 02.12.2023
  + Звершення опрацювання теми: 12.12.2023
* Тема №2 Класи. Шаблони класів.
  + Джерела Інформації:
    - Відео.

<https://www.youtube.com/watch?v=ZbsukxxV5_Q&t=1282s&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>

<https://www.youtube.com/watch?v=a5jigk0Tw50&t=282s&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>

<https://www.youtube.com/watch?v=0YWOW8BezI4&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>

* + Що опрацьовано:
    - Класи. Шаблони класів
    - Специфікатори доступу (private, public)
    - Конструктори та деструктори
  + Статус: Ознайомлений
  + Початок опрацювання теми: 04.12.2023
  + Звершення опрацювання теми: 12.12.2023

# **Виконання роботи:**

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

Завдання №1 VNS Lab 10

* Варіант 15

Записи в лінійному списку містять ключове поле типу \*char (рядок

символів). Сформувати двонаправлений список. Знищити К елементів з

кінця списку. Додати елемент після елемента із заданим ключем.

* Важливі деталі для врахування в імплементації програми

написав фунцції для запису у файл, відновлення з файлу.

Завдання №2 Algotester Lab 5

* Варіант 3
* Деталі завдання

У вас є карта гори розмiром 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 пiк гори мав найбiльше число.

Клiтинкi якi мають сумiжну сторону з вершиною мають висоту на один меншу, сумiжнi з

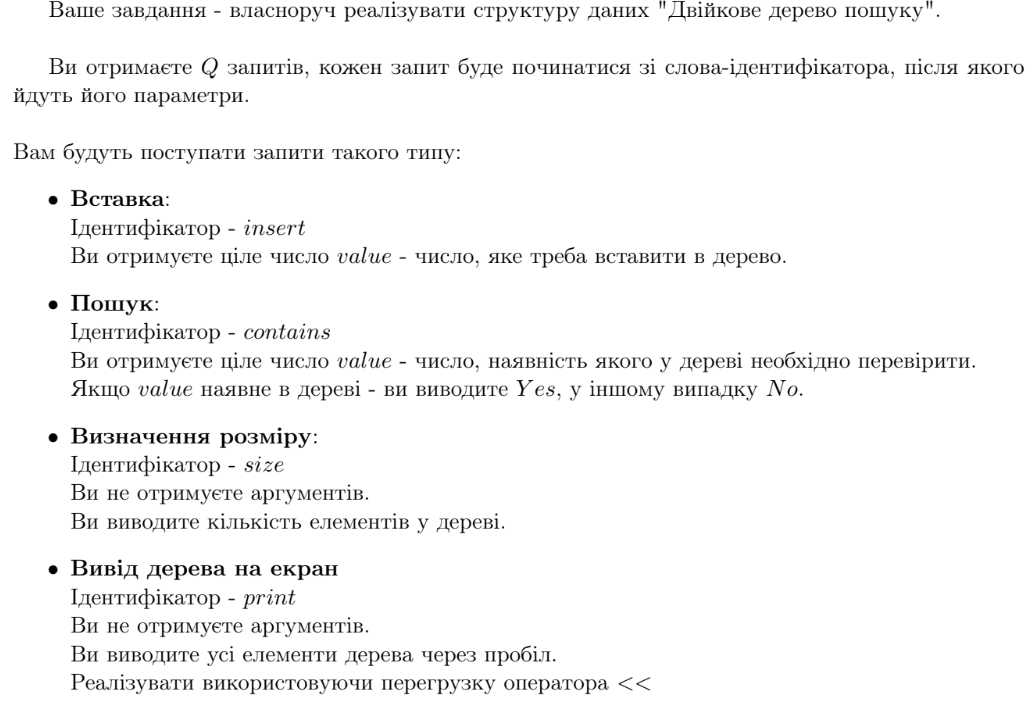
ними i не розфарбованi мають ще на 1 меншу висоту i так далi.

* Важливі деталі для врахування в імплементації програми

Використовую двовимірний масив чисел для відображення карти. Шукав відстань до піку від кожного кута.

Завдання №3 Algotester Lab 7-8

* Варіант 3
* Деталі завдання



* Важливі деталі для врахування в імплементації програми

Реалізовую через шаблон класу

Завдання №4 Class Practice Work

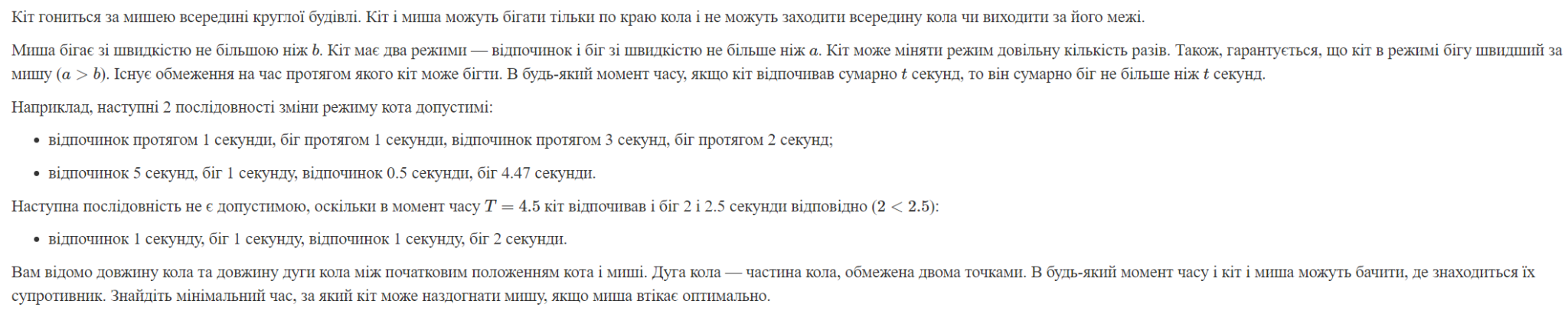
* Без варіанту
* Деталі завдання

Задача №1 - Реверс списку (Reverse list). Задача №2 - Порівняння списків. Задача №3 – Додавання великих чисел. Задача №4 - Віддзеркалення дерева. Задача №5 - Записати кожному батьківському вузлу суму підвузлів

* Важливі деталі для врахування в імплементації програми

Здійснюю обхід дерева двома способами використовуючи рекурсію

Завдання №4 Self Practice Work

* Погоня в колі
* 
* Деталі завдання

Реалізація черги

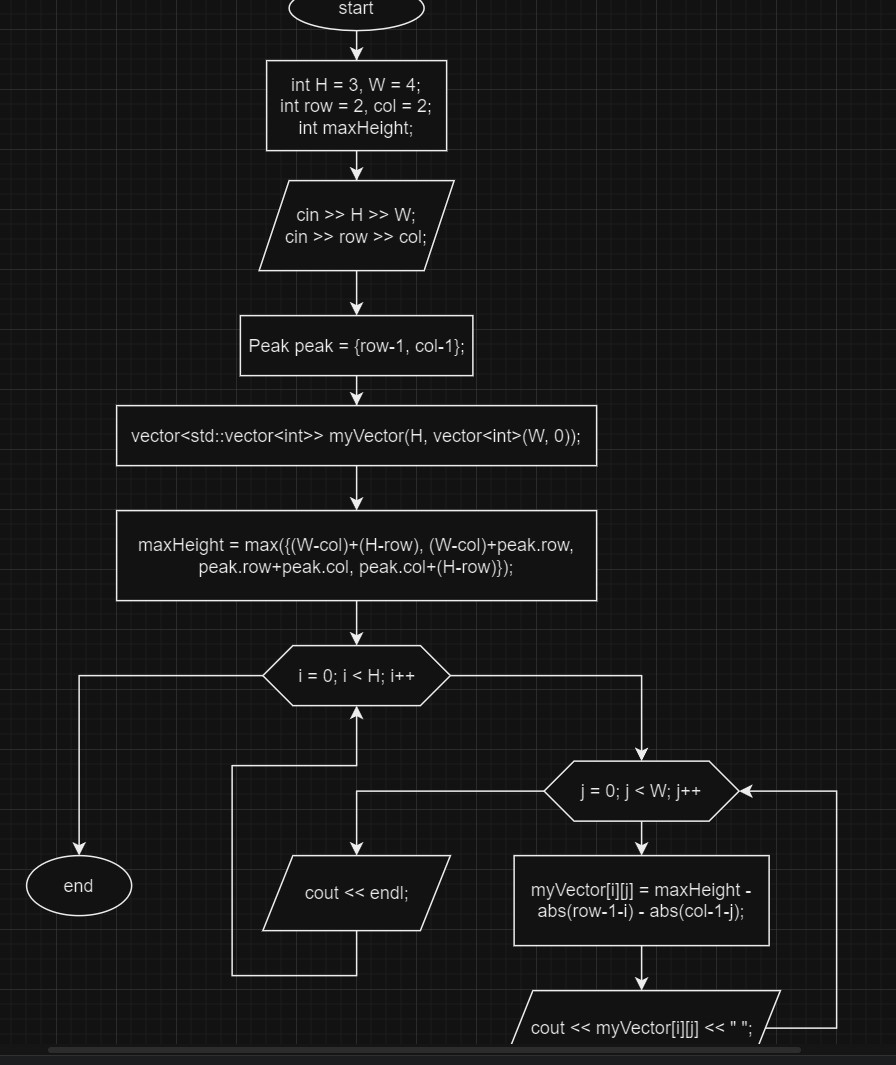
* Важливі деталі для врахування в імплементації програми

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

Програма №1 VNS Lab 10

* Планований час на реалізацію: 2 год

Програма №2 Algotester Lab 5

* Планований час на реалізацію: 1 год
* 

*Зображення №1. Блок-схема до програми №4 algotester Lab 5*

Програма №3 Algotester Lab 7-8

* Планований час на реалізацію: 2 год
* Важливі деталі для врахування в імплементації

Програма №4 Practice Work

* Планований час на реалізацію: 2,5 год

Програма №4 Self Practice Work

* Планований час на реалізацію: 30 хв
* Важливі деталі для врахування в імплементації

## **3. Конфігурація середовища до виконання завдань:**

Використано налаштування з попередніх лабораторних робіт

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

Посилання на файл програми у пул-запиті GitHub: <https://github.com/artificial-intelligence-department/ai_programming_playground/pull/808>

Завдання №1 VNS Lab 10

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

struct Node {

string data;

Node\* next;

Node\* prev;

};

class DoubleLinkedList {

Node\* head;

Node\* tail;

public:

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

void add(string value) {

Node\* newNode = new Node{value, nullptr, tail};

if (head == nullptr) {

head = tail = newNode;

}

else {

tail->next = newNode; *//(\*tail).next = newNode;*

tail = newNode;

}

}

void add\_after(string value, string after) {

Node\* current = tail;

while (current != nullptr) {

if (current->data == after) {

Node\* newNode = new Node{value, current->next, current};

if (current->next != nullptr) {

current->next->prev = newNode;

}

current->next = newNode;

if (current == tail) {

tail = newNode;

}

return;

}

current = current->prev;

}

}

void print(){

Node\* current = head;

if (current == nullptr) {

cout << "Empty list\n";

return;

}

while (current != nullptr) {

cout << current->data << ' ';

current = current->next;

}

cout << '\n';

}

void deleteList() {

Node\* current = head;

while (current != nullptr) {

Node\* next = current->next;

delete current;

current = next;

}

head = nullptr;

}

void deleteNode(int amount) {

Node\* current = tail;

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

if (current == nullptr) {

return;

}

Node\* prev = current->prev;

if (prev != nullptr) {

prev->next = nullptr;

}

delete current;

current = prev;

}

tail = current;

}

void copy\_to\_file(string filename){

ofstream file(filename);

Node\* current = head;

while(current != nullptr){

file << current->data << endl;

current = current->next;

}

file.close();

}

void copy\_from\_file(string filename){

ifstream file(filename);

string value;

while(file >> value){

add(value);

}

file.close();

}

};

int main() {

DoubleLinkedList list;

list.add("ya");

list.add("tu");

list.add("te");

list.add("tk");

list.add("tg");

list.add("ge");

list.add("uu");

list.add("mm");

list.print();

list.deleteNode(4);

list.print();

list.add\_after("ro", "tk");

list.print();

list.copy\_to\_file("file.txt");

list.deleteList();

list.print();

list.copy\_from\_file("file.txt");

list.print();

return 0;

}

Завдання №2 Algotester Lab 5

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

struct Peak{

int row;

int col;

};

int main() {

int H = 3, W = 4;

int row = 2, col = 2;

int maxHeight;

cin >> H >> W;

cin >> row >> col;

Peak peak = {row-1, col-1};

vector<std::vector<int>> myVector(H, vector<int>(W, 0));

maxHeight = max({(W-col)+(H-row), (W-col)+peak.row, peak.row+peak.col, peak.col+(H-row)});

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

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

myVector[i][j] = maxHeight - abs(row-1-i) - abs(col-1-j);

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

}

cout << endl;

}

return 0;

}

Завдання №3 Algotester Lab 7-8

#include <iostream>

#include <string>

using namespace std;

template <typename T>

class BinaryTree{

private:

struct Node {

T data;

Node\* left;

Node\* right;

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

};

Node\* root;

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

if (node == nullptr) {

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;

}

void print(Node\* node) {

if (node != nullptr) {

print(node->left);

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

print(node->right);

}

}

bool search(Node\* node, T value) {

if (node == nullptr) {

return false;

}

if (value == node->data) {

return true;

}

if (value < node->data) {

return search(node->left, value);

} else {

return search(node->right, value);

}

}

int size(Node\* node) {

if (node == nullptr) {

return 0;

}

return 1 + size(node->left) + size(node->right);

}

public:

BinaryTree() : root(nullptr) {}

void insert(T value) {

root = insert(root, value);

}

void print() {

print(root);

cout << endl;

}

bool search(T value) {

return search(root, value);

}

int size() {

return size(root);

}

};

int main() {

BinaryTree<int> myTree;

int Q, K;

string input;

cin >> Q;

cout << endl;

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

cin >> input;

if(input == "insert"){

cin >> K;

myTree.insert(K);

}

else if(input == "print"){

myTree.print();

}

else if(input == "contains"){

cin >> K;

if(myTree.search(K)){

cout << "Yes" << endl;

}

else{

cout << "No" << endl;

}

}

else if(input == "size"){

cout << myTree.size() << endl;

}

}

return 0;

}

Завдання №4 Class Practice Work

4.1

#include <iostream>

struct Node {

int data;

Node\* next;

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

};

class LinkedList {

Node\* head;

public:

LinkedList() : head(nullptr) {}

void insert(int data) {

Node\* newNode = new Node(data);

if (head == nullptr) {

head = newNode;

} else {

Node\* current = head;

while (current->next != nullptr) {

current = current->next;

}

current->next = newNode;

}

}

void print() {

Node\* current = head;

while (current != nullptr) {

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

current = current->next;

}

std::cout << std::endl;

}

void reverse() {

Node\* current = head;

Node\* prev = nullptr;

Node\* next = nullptr;

while (current != nullptr) {

next = current->next;

current->next = prev;

prev = current;

current = next;

}

head = prev;

}

};

int main() {

LinkedList list;

list.insert(1);

list.insert(2);

list.insert(3);

list.insert(4);

list.print();

list.reverse();

list.print();

return 0;

}

4.2

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

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

};

class LinkedList {

Node\* head;

public:

LinkedList() : head(nullptr) {}

void insert(int data) {

Node\* newNode = new Node(data);

if (head == nullptr) {

head = newNode;

} else {

Node\* current = head;

while (current->next != nullptr) {

current = current->next;

}

current->next = newNode;

}

}

void print() {

Node\* current = head;

while (current != nullptr) {

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

current = current->next;

}

std::cout << std::endl;

}

bool compare(Node\* node1, Node\* node2) {

if (node1 == nullptr && node2 == nullptr) {

return true;

}

if (node1 == nullptr || node2 == nullptr) {

return false;

}

return node1->data == node2->data && compare(node1->next, node2->next);

}

bool compareLists(LinkedList& list1, LinkedList& list2) {

return compare(list1.head, list2.head);

}

};

int main() {

LinkedList list;

LinkedList list2;

list.insert(1);

list.insert(2);

list.insert(3);

list.insert(4);

list2.insert(1);

list2.insert(2);

list2.insert(3);

list2.insert(5);

cout << (list.compareLists(list, list2) ? "true" : "false") << endl;

list.print();

return 0;

}

4.3

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

struct Node {

char data;

Node\* next;

Node\* prev;

};

class DoubleLinkedList {

Node\* head;

Node\* tail;

public:

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

void add(char value) {

Node\* newNode = new Node{value, nullptr, tail};

if (head == nullptr) {

head = tail = newNode;

}

else {

tail->next = newNode; *//(\*tail).next = newNode;*

tail = newNode;

}

}

void add\_from\_string(string value) {

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

add(value[i]);

}

reverse();

}

void print() {

Node\* current = tail;

if (current == nullptr) {

cout << "Empty list\n";

return;

}

while (current != nullptr) {

cout << current->data;

current = current->prev;

}

cout << '\n';

}

void reverse() {

Node\* current = head;

Node\* temp = nullptr;

while (current != nullptr) {

temp = current->prev;

current->prev = current->next;

current->next = temp;

current = current->prev;

}

temp = head;

head = tail;

tail = temp;

}

void sum\_of\_lists(DoubleLinkedList& list1, DoubleLinkedList& list2, DoubleLinkedList& output) {

Node\* current1 = list1.head;

Node\* current2 = list2.head;

int carry = 0;

while (current1 != nullptr || current2 != nullptr) {

int sum = carry;

if (current1 != nullptr) {

sum += current1->data - '0'; *// '2' - '0' = 2 (50-48)*

current1 = current1->next;

}

if (current2 != nullptr) {

sum += current2->data - '0';

current2 = current2->next;

}

carry = sum / 10;

output.add(sum % 10 + '0');

}

if (carry > 0) {

output.add(carry + '0');

}

}

};

int main() {

DoubleLinkedList list1;

DoubleLinkedList list2;

DoubleLinkedList output;

list1.add\_from\_string("500");

list2.add\_from\_string("500");

list1.print();

list2.print();

list1.sum\_of\_lists(list1, list2, output);

output.print();

return 0;

}

*4.4*

*#include <iostream>*

*#include <string>*

*using namespace std;*

*template <typename T>*

*class BinaryTree{*

*private:*

*struct Node {*

*T data;*

*Node\* left;*

*Node\* right;*

*Node(T data) : data(data), left(nullptr), right(nullptr) {}*

*};*

*Node\* root;*

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

*if (node == nullptr) {*

*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;*

*}*

*void print(Node\* node) {*

*if (node != nullptr) {*

*print(node->left);*

*cout << node->data << " ";*

*print(node->right);*

*}*

*}*

*Node\* flipAndCopy(Node\* node) {*

*if (node == nullptr) {*

*return nullptr;*

*}*

*Node\* newNode = new Node(node->data);*

*newNode->left = flipAndCopy(node->right);*

*newNode->right = flipAndCopy(node->left);*

*return newNode;*

*}*

*public:*

*BinaryTree() : root(nullptr) {}*

*void insert(T value) {*

*root = insert(root, value);*

*}*

*void print() {*

*print(root);*

*cout << endl;*

*}*

*BinaryTree flipTree() {*

*BinaryTree newTree;*

*newTree.root = flipAndCopy(root);*

*return newTree;*

*}*

*};*

*int main() {*

*BinaryTree<int> myTree;*

*BinaryTree<int> myTree2;*

*myTree.insert(15);*

*myTree.insert(10);*

*myTree.insert(8);*

*myTree.insert(55);*

*myTree.insert(30);*

*myTree.insert(3);*

*myTree.insert(1);*

*myTree2 = myTree.flipTree();*

*myTree.print();*

*myTree2.print();*

*return 0;*

*}*

4.5

#include <iostream>

#include <string>

using namespace std;

template <typename T>

class BinaryTree{

private:

struct Node {

T data;

Node\* left;

Node\* right;

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

};

Node\* root;

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

if (node == nullptr) {

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;

}

void print(Node\* node) {

if (node != nullptr) {

print(node->left);

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

print(node->right);

}

}

void tree\_sum(Node\* node) {

if (node == nullptr) {

return;

}

*// First recur for left and right subtrees*

tree\_sum(node->left);

tree\_sum(node->right);

*// Update the value of this node as the sum of its children*

if (node->left != nullptr) {

node->data = node->left->data;

}

if (node->right != nullptr) {

node->data += node->right->data;

}

}

public:

BinaryTree() : root(nullptr) {}

void insert(T value) {

root = insert(root, value);

}

void print() {

print(root);

cout << endl;

}

void tree\_sum() {

tree\_sum(root);

}

};

int main() {

BinaryTree<int> myTree;

myTree.insert(15);

myTree.insert(10);

myTree.insert(8);

myTree.insert(55);

myTree.insert(30);

myTree.print();

myTree.tree\_sum();

cout << endl;

myTree.print();

return 0;

}

Завдання №5 Self Practice Work

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

double minTimeToCatchMouse(double l, double d, double a, double b) {

double time1 = 1e9, time2 = 1e9;

if (0.5 \* a > b) {

time1 = d / (0.5 \* a - b);

}

time2 = 2 \* (0.5 \* l / (a - b));

return min(time1, time2);

}

int main() {

double l, d, a, b;

cin >> l >> d >> a >> b;

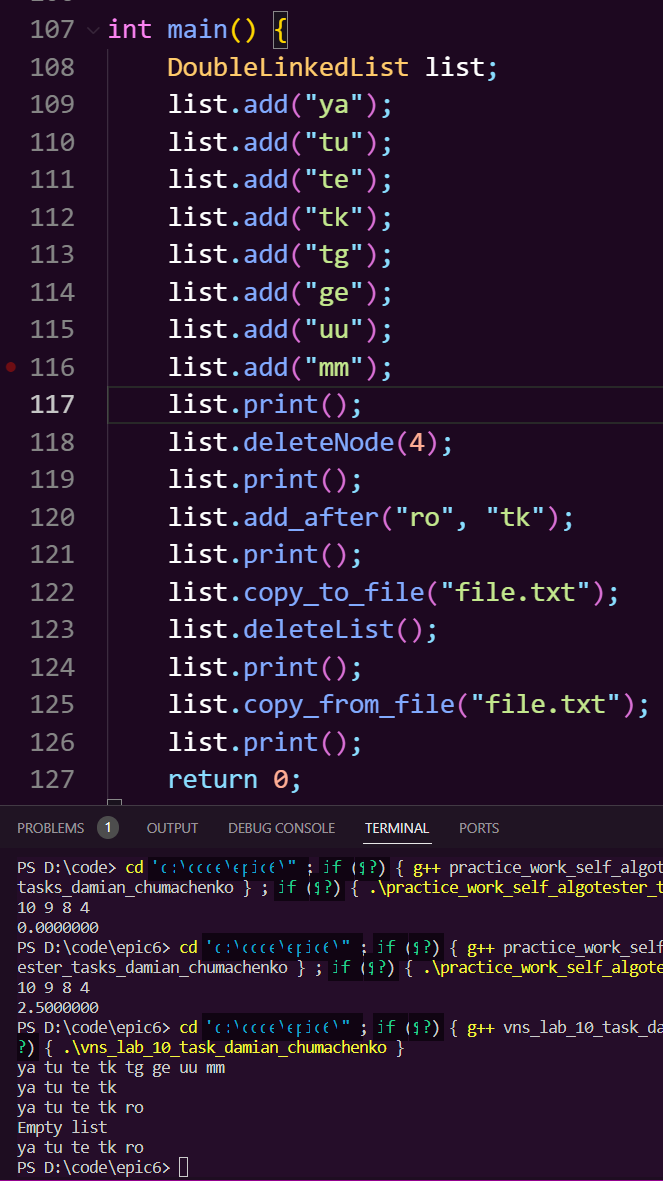
cout << fixed << setprecision(7) << minTimeToCatchMouse(l, d, a, b) << endl;

return 0;

}

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

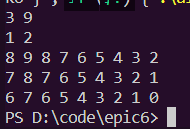
Завдання №1 VNS Lab 10



Затрачений час: 2 год

*Зображення №2. Тест до програми* №1 VNS Lab 10

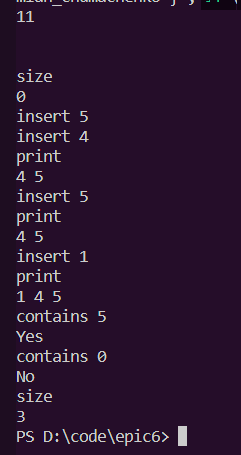
Завдання №2 Algotester Lab 5



Затрачений час: 1 год

*Зображення №3. Тест до програми* №2 Algotester lab 5

Завдання №3 Algotester Lab 7-8

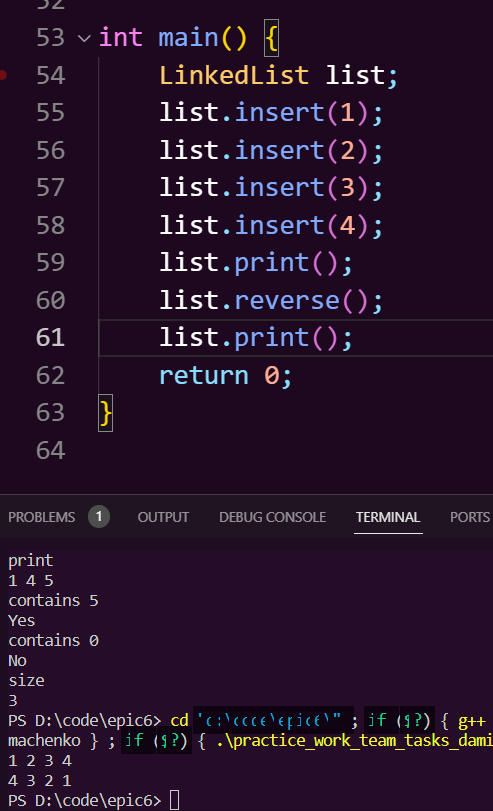


Затрачений час: 2.5 год

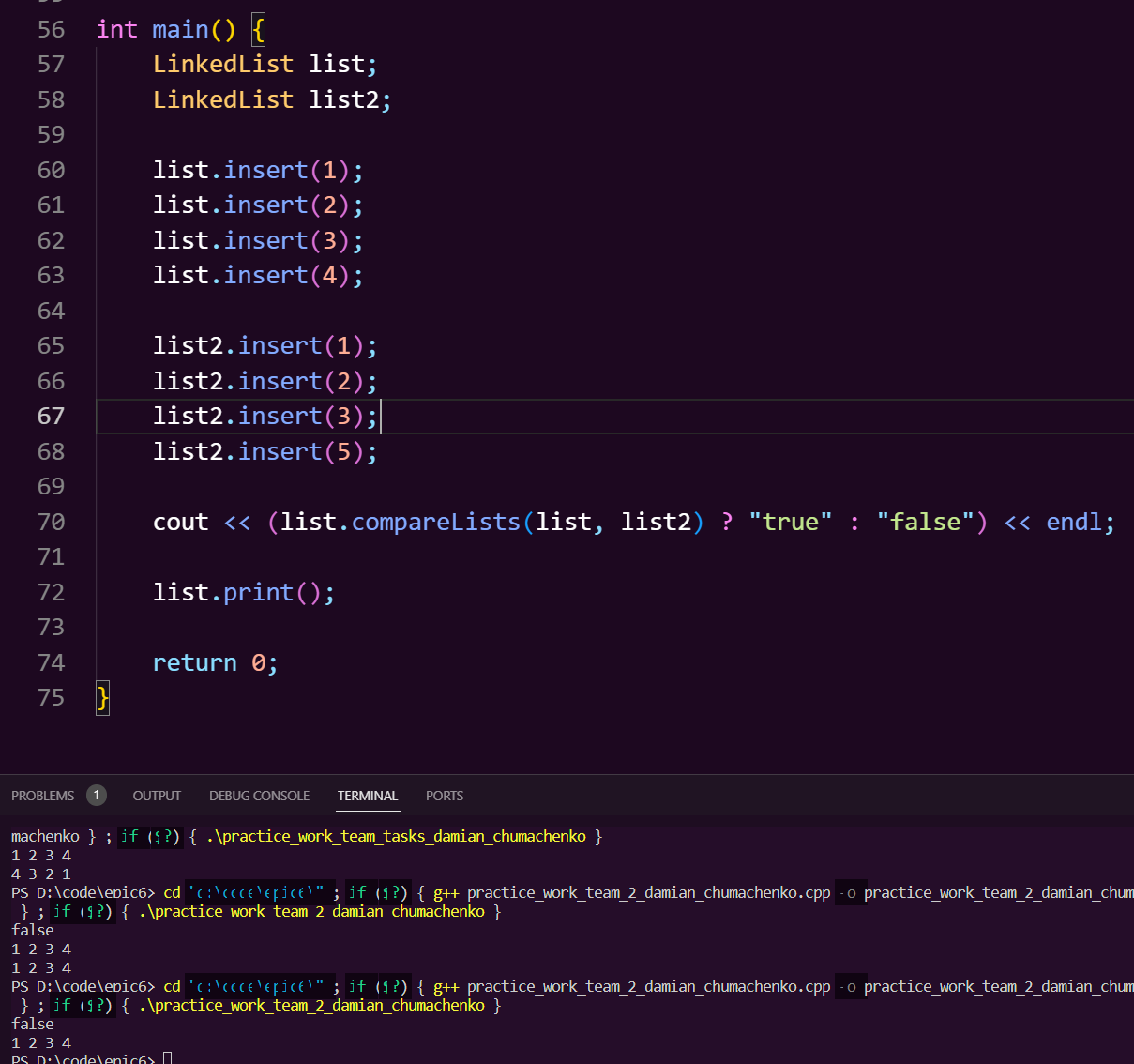
*Зображення №4. Тест до програми* №2 Algotester lab 7-8

Завдання №4 Class Practice Work

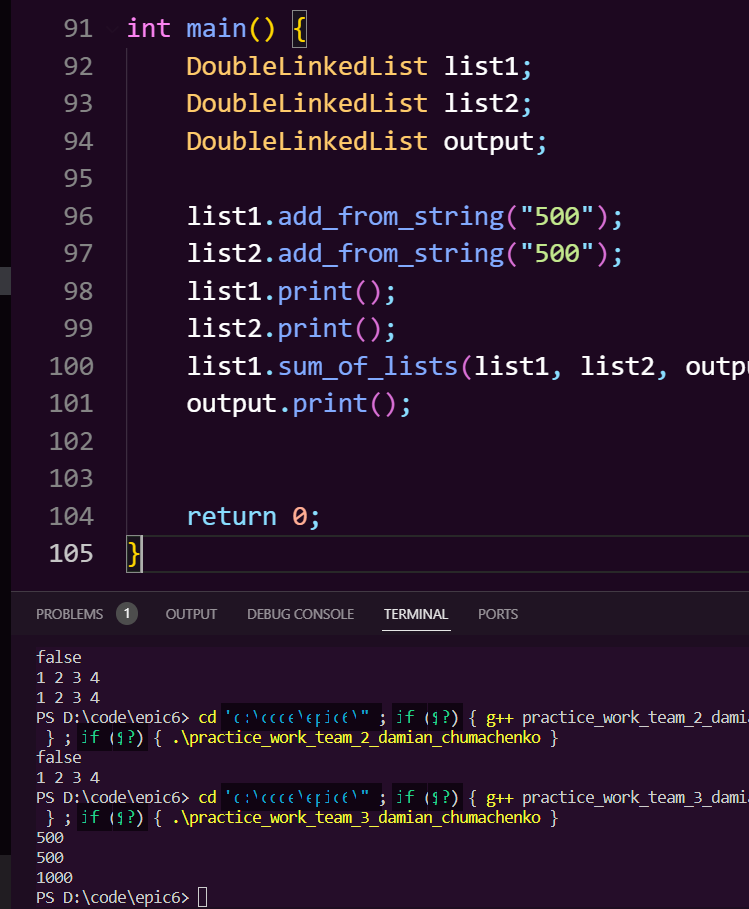
Затрачений час: 4 год



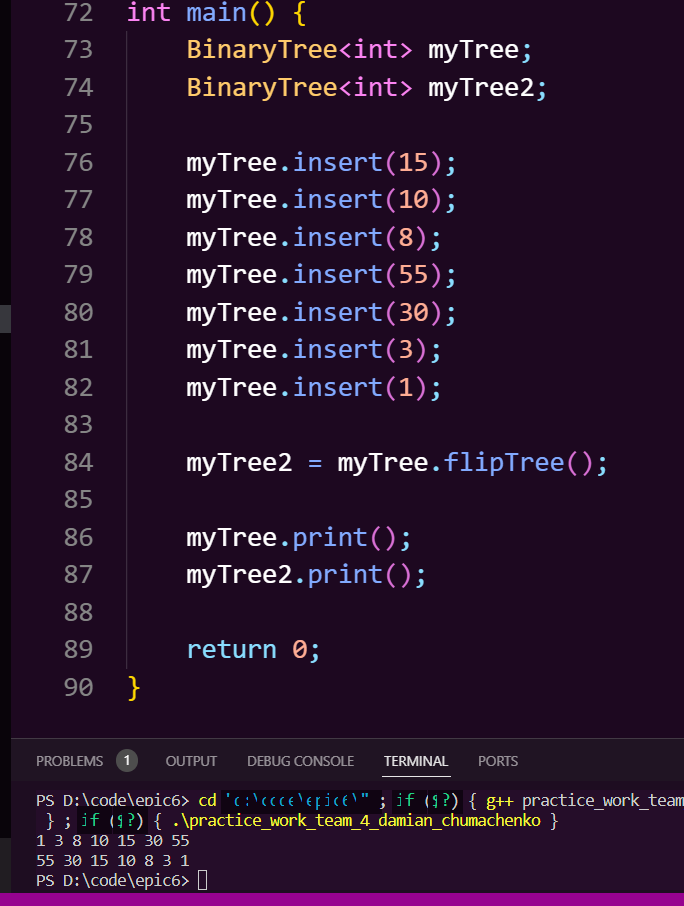
*Зображення №5. Тест до програми* class practice 4.1



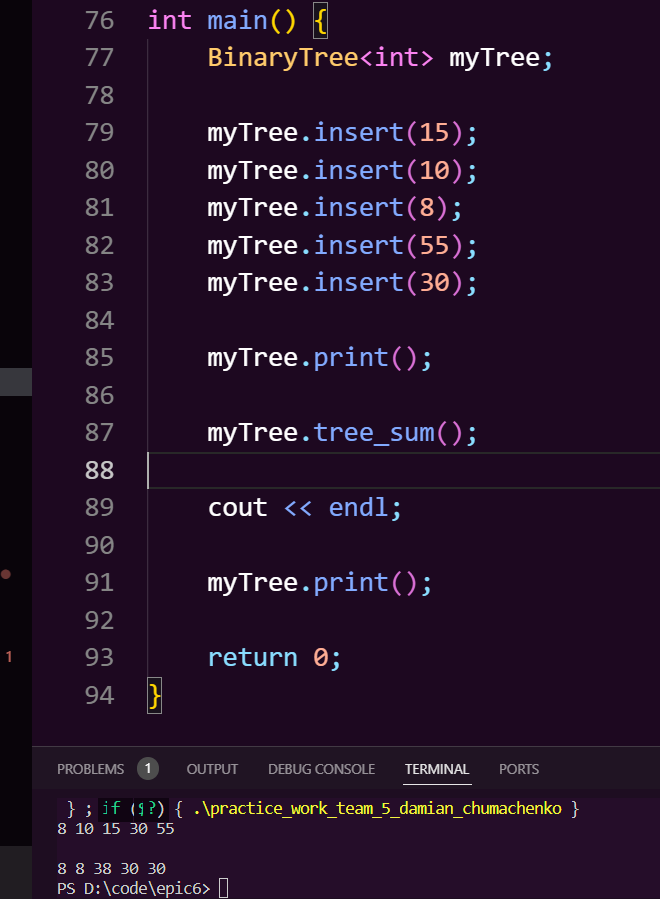
*Зображення №6. Тест до програми* class practice 4.2



*Зображення №7. Тест до програми* class practice 4.3



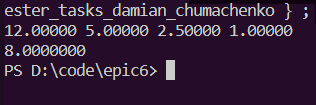
*Зображення №8. Тест до програми* class practice 4.4



*Зображення №9. Тест до програми* class practice 4.5

Завдання №5 Self Practice Work

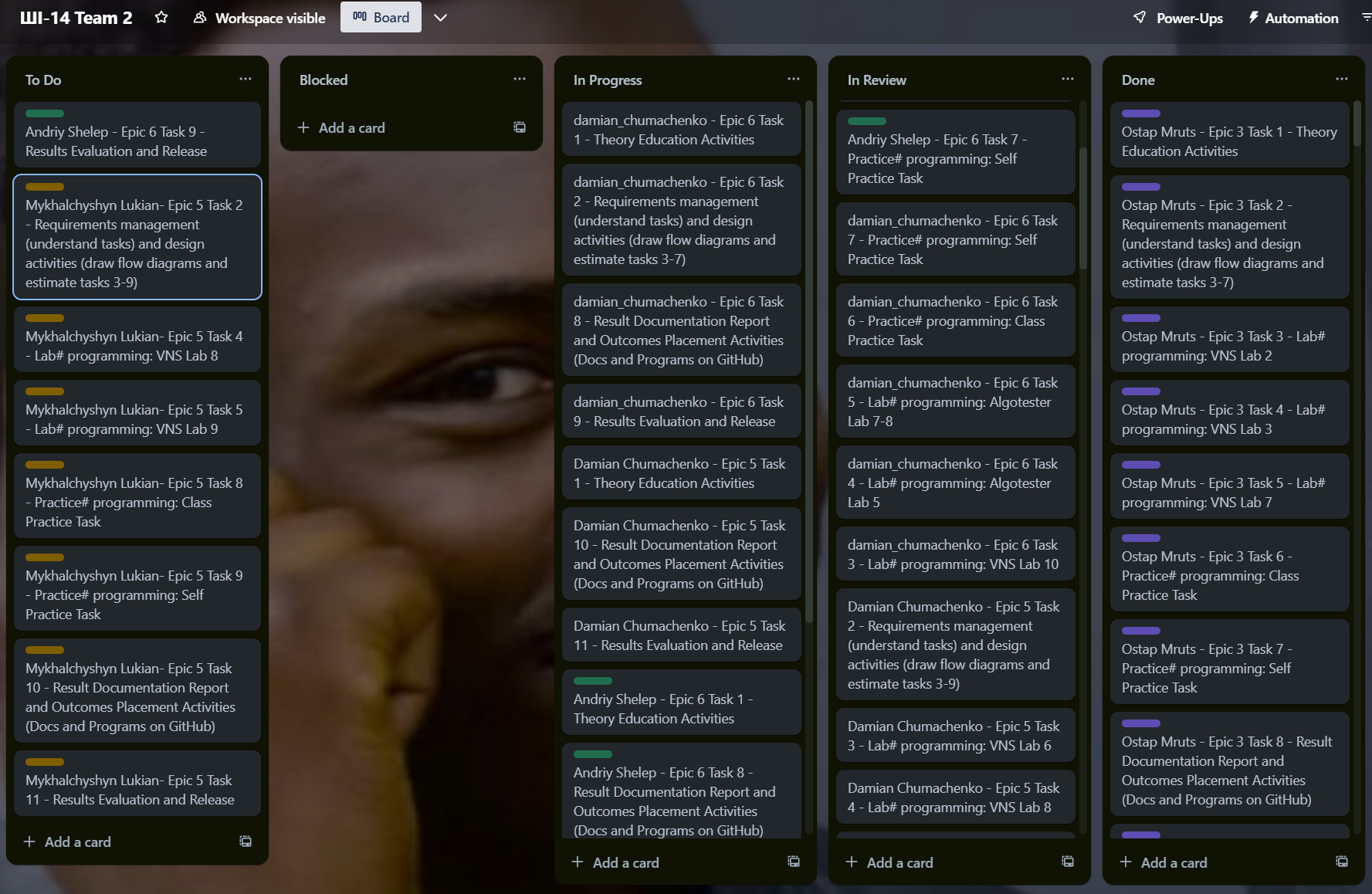
Затрачений час: 1 год



## **6. Кооперація з командою:**

* Скрін зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло





# **Висновки:**

Опрацьовано динамічні структури даних та реалізовано деякі з них. Ознайомився з деревами, чергою, двозв’язним і однозв’язним списками та іншими структурами даних. Використано динамічні структури даних у виконанні завдань з лабораторної роботи.