Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

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

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

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

до:

ВНС Лабораторної Роботи № 10

Алготестер Лабораторної Роботи № 5

Алготестер Лабораторної Роботи № 7-8

Практичних Робіт № 6

**Виконала:**

Студентка групи ШІ-12

Виклюк Яна Ярославівна

# **Тема роботи:**

Ознайомлення з динамічними структурами (Черга, Стек, Списки, Дерево), алгоритмами обробки динамічних структур.

# **Мета роботи:**

Ретельно освоїти динамічні структури даних, такі як Черга, Стек, Списки та Дерево, а також алгоритми, що застосовуються для їх обробки. Отримати навички використання цих структур для ефективного управління даними в програмах.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Динамічні структури даних(списки, дерево).
* Тема №2: Динамічні структури даних (черга, стек).

1. Індивідуальний план опрацювання теорії:

* Тема №1: Динамічні структури (списки, дерево).
  + Джерела Інформації:
    - <https://allref.com.ua/uk/skachaty/Dinamichni_strukturi_danih_%28S++%29>
    - <https://erudyt.net/navchalni-predmety/informatika/prohramuvannya/linijnyj-odnozvyaznyj-spysok.html>
    - <https://www.bestprog.net/uk/2022/02/16/c-linear-doubly-linked-bidirectional-list-general-concepts-ua/>
    - <https://www.bestprog.net/uk/2023/08/01/c-binary-search-tree-binarytree-class-integers-ua/>
  + Що опрацьовано:
    - Опрацьовано динамічні структури даних, а саме, списки (одно- та двонапрямлені) та бінарні дерева, особливості роботи з ними.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 08.12
* Звершення опрацювання теми: 09.12
* Тема №2 : Динамічні структури (черга, стек).
  + Джерела Інформації
  + <https://acode.com.ua/urok-111-stek-i-kupa/>
  + <https://uk.wikipedia.org/wiki/Черга_(структура_даних)>
  + Що опрацьовано:
    - Опрацьовано динамічні структури даних, а саме, чергу та стек.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 09.12
  + Звершення опрацювання теми: 10.12

# **Виконання роботи:**

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

Завдання №1 VNS Lab 10

* Варіант №8
* Деталі завдання
* Записи в лінійному списку містять ключове поле типу int. Сформувати
* двонаправлений список. Знищити з нього елемент після елемента із заданим
* номером, додати К елементів у початок списку, виконати друк списку, записати його у файл, видалити список й виконати друк, відновити список з файлу та виконати його друк.
* Важливі деталі для врахування в імплементації програми
* Уважно працювати з двонаправленим списком, враховуючи, що потрібно змінювати вказівник не тільки на наступний, а й на попередній елемент при видаленні та додаванні якихось елементів.

Завдання №2 Algotester Lab 5

* Варіант №3
* Умова
* 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.
* Клітинкі 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.
* Вхідні дані
* У першому рядку 2 числа N та M - розміри карти
* у другому рядку 2 числа x та y - координати піку гори
* Деталі завдання
* Вивести N рядків по M елементів в рядку через пробіл - висоти карти.
* Важливі деталі для врахування в імплементації програми
* Врахувати, що відстань до певної точки дорівнює сумі модулів різниць координати вершини х та заданої точки і вершини у та заданої точки.

Завдання №3 Algotester Lab 7-8

* Варіант №1
* Умова
* Ваше завдання - власноруч реалізувати структуру даних "Двозв’язний список".  
  Ви отримаєте Q запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його аргументи.  
  Вам будуть поступати запити такого типу:
* Вставка:  
  Ідентифікатор - insert  
  Ви отримуєте ціле число index елемента, на місце якого робити вставку.  
  Після цього в наступному рядку рядку написане число N - розмір списку, який треба вставити.  
  У третьому рядку N цілих чисел - список, який треба вставити на позицію index.
* Видалення:  
  Ідентифікатор - erase  
  Ви отримуєте 2 цілих числа - index, індекс елемента, з якого почати видалення та n - кількість елементів, яку треба видалити.
* Визначення розміру:  
  Ідентифікатор - size  
  Ви не отримуєте аргументів.  
  Ви виводите кількість елементів у списку.
* Отримання значення i-го елементу  
  Ідентифікатор - get  
  Ви отримуєте ціле число - index, індекс елемента.  
  Ви виводите значення елемента за індексом.
* Модифікація значення i-го елементу  
  Ідентифікатор - set  
  Ви отримуєте 2 цілих числа - індекс елемента, який треба змінити, та його нове значення.
* Вивід списку на екран  
  Ідентифікатор - print  
  Ви не отримуєте аргументів.  
  Ви виводите усі елементи списку через пробіл.  
  Реалізувати використовуючи перегрузку оператора <<
* Вхідні дані
* Ціле число Q - кількість запитів.  
  У наступних рядках Q запитів у зазначеному в умові форматі.
* Деталі завдання
* Вивести відповіді на запити у зазначеному в умові форматі.
* Важливі деталі для врахування в імплементації програми
* Зрозуміти принципи та особливості роботи двозвʼязного списку для правильної реалізації.

Завдання №4 Class Practice Work 1

* Варіант відсутній
* Деталі завдання
* Реалізувати метод реверсу зв’язного списку.
* Важливі деталі для врахування в імплементації програми
* Врахувати використання цілочисельних значень в списку. Реалізувати допоміжний метод виведення вхідного і обернутого списків.

Завдання №5 Class Practice Work 2

* Варіант відсутній
* Деталі завдання
* Реалізувати метод порівняння зв’язних списків.
* Важливі деталі для врахування в імплементації програми
* Врахувати використання цілочисельних значень 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 №6 Class Practice Work 3

* Варіант відсутній
* Деталі завдання
* Реалізувати метод додавання великих чисел за допомогою зв’язного списку.
* Важливі деталі для врахування в імплементації програми
* Врахувати використання цифр 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 №7 Class Practice Work 4

* Варіант завдання відсутній
* Деталі завдання
* Реалізувати метод віддзеркалення дерева.
* Важливі деталі для врахування в імплементації програми
* Врахувати використання цілочисельних значень в списку, 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 №8 Class Practice Work 5

* Варіант завдання відсутній
* Деталі завдання
* Реалізувати метод запису кожному батьківському вузлу дерева суму його підвузлів.
* Важливі деталі для врахування в імплементації програми
* Врахувати використання цілочисельних значень в списку, 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.

Завдання №9 Self Practice Work (Algotester lab3v1)

* Варіант завдання №1
* Умова
* Ви з’явились у світі під назвою Атод посеред Пустелі Безправ’я. Так сталось, що Ви попали саме в той час і місце, де ведеться битва між чаклункою Ліною і темними силами, які хочуть знищити цей світ. На жаль, трапилась халепа, бо деякі слова із книги чар були пошкоджені під час битви. Одне таке слово можна відновити виконавши ритуал зцілення над пошкодженими буквами. Ритуал зцілення можна виконати на всіх підрядрозташованих пошкоджених буквах. Вам не залишається нічого іншого як допомогти Ліні відновити ці слова і сказати скільки мінімально треба провести таких ритуалів, щоб прочитати одне з наймогутніших у цьому світі заклять - Поневолення Дракона!
* Вхідні дані
* У першому рядку N - кількість рядків у заклятті.
* В наступних N рядкях - набір слів w1,…,wM, розділених пробілами, де кожне слово може містити малі латинські літери та символ #, який позначає пошкоджену букву.
* Деталі завдання
* Вивести єдине ціле число — мінімальна кількість ритуалів, які потрібно провести, щоб відновити закляття.
* Важливі деталі для врахування в імплементації програми
* Врахувати, що якщо # іде під ряд, то це буде вважатись одним ритуалом.

Завдання №10 Self Practice Work (Algotester lab6v3)

* Варіант №3
* Умова
* У Клінта в черговий раз виключилось світло і йому немає чим зайнятися. Так як навіть це не заставить його подивитися збережені відео про програмування на ютубі - він вирішив придумати свою гру на основі судоку.
* Гра виглядає так:
* Є поле розміром N×N, в якому частина клітинок заповнена цифрами, а частина клітинок пусті (позначаються нулем). Також у нього є Q пар координат X та Y.
* Завданням гри є написати до кожної координати скільки чисел туди можна вписати (якщо вона пуста) і які це числа (обов’язково в посортовані по зростанню!). В клітинку можна вписати лише ті числа, які не зустрічаються в рядку та стовбці, які перетинаються у цій клітинці.
* Під час гри поле не міняється!
* Також необовязково, щоб це було валідне судоку! Якщо є клітинка, в яку не можна вписати ніяку цифру - виведіть 0.
* Також допускаються рядки та стовпці, в яких цифра записана кілька разів.
* Вхідні дані
* У першому рядку ціле число N - розмір поля для гри
* У N наступних рядках стрічка rowi яка складається з N цифер - i-й рядок.
* Ціле число Q - кількість запитань
* У наступних Q рядках 2 цілих числа xj,yj - координати клітинок j-го запитання
* Деталі завдання
* Вивести Q разів відповідь у наступному форматі:
* Натуральне число M - кількість цифр, які можна вписати в клітинку
* M цифер розділених пробілом - можливі цифри.
* Важливі деталі для врахування в імплементації програми
* Перевести кожен елемент стрічки, котра задається одразу у число, як тільки вона вводиться.

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

Програма №1 VNS Lab 10

* Планований час на реалізацію 2 год
* Важливі деталі для врахування в імплементації
* Уважно працювати з двонаправленим списком, враховуючи, що потрібно змінювати вказівник не тільки на наступний, а й на попередній елемент при видаленні та додаванні якихось елементів.

Програма №2 Algotester Lab 5

* Блок-схема

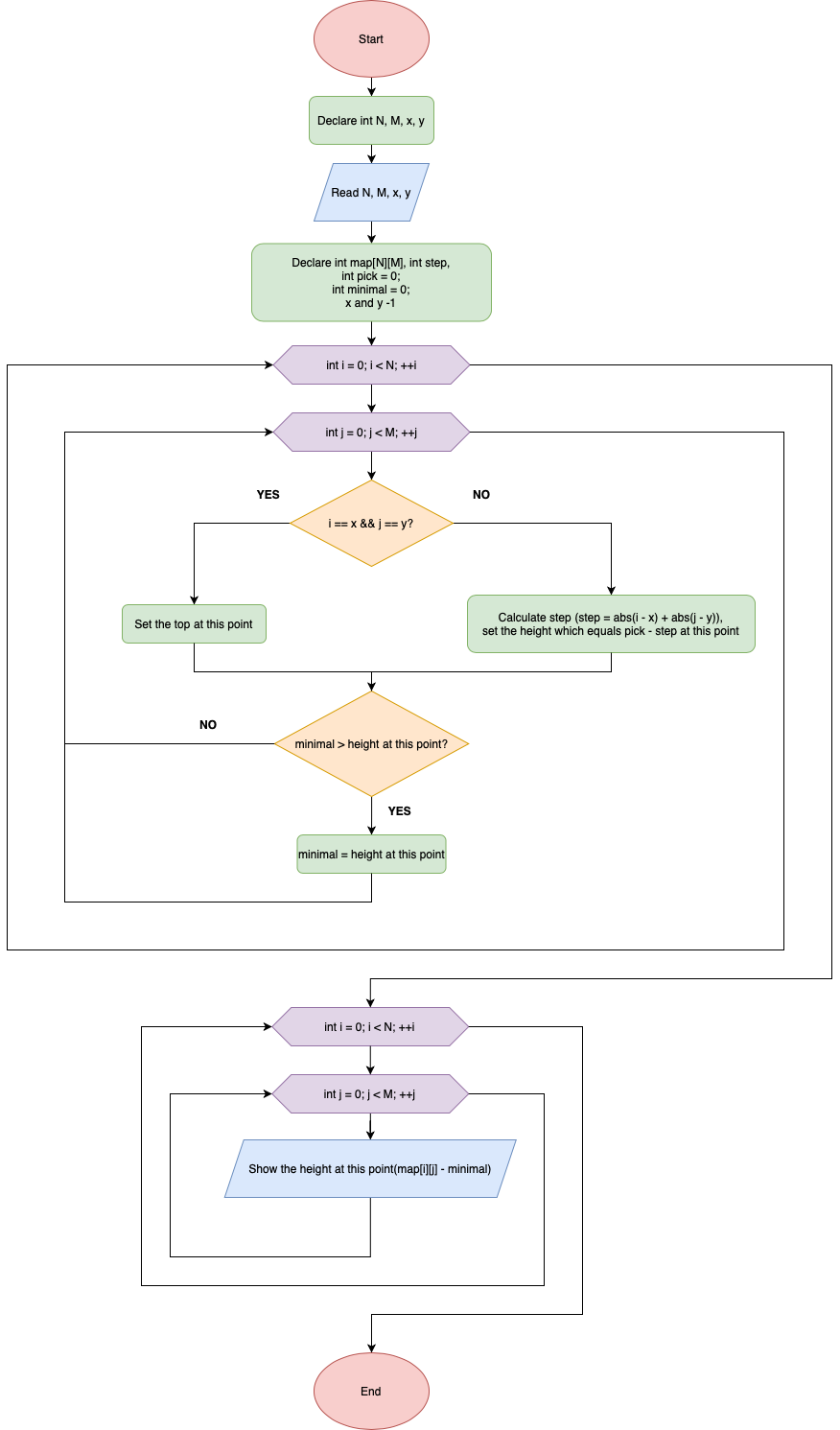


Рисунок 1 Блок-схема до програми №2

* Планований час на реалізацію 90 хв
* Важливі деталі для врахування в імплементації
* Врахувати, що відстань до певної точки дорівнює сумі модулів різниць координати вершини х та заданої точки і вершини у та заданої точки.

Програма №3 Algotester Lab 7-8

* Планований час на реалізацію 2 год
* Важливі деталі для врахування в імплементації
* Зрозуміти принципи та особливості роботи двозвʼязного списку для правильної реалізації.

Програма №4 Class Practice Work 1

* Планований час на реалізацію 2 год
* Важливі деталі для врахування в імплементації
* Врахувати використання цілочисельних значень в списку. Реалізувати допоміжний метод виведення вхідного і обернутого списків.

Програма №5 Class Practice Work 2

* Планований час на реалізацію 90 хв
* Важливі деталі для врахування в імплементації
* Врахувати використання цілочисельних значень 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.

Програма №6 Class Practice Work 3

* Планований час на реалізацію 2 год
* Важливі деталі для врахування в імплементації
* Врахувати використання цілочисельних значень в списку. Реалізувати допоміжний метод виведення вхідного і обернутого списків.

Програма №7 Class Practice Work 4

* Планований час на реалізацію 2 год
* Важливі деталі для врахування в імплементації
* Врахувати використання цілочисельних значень в списку, 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.

Програма №8 Class Practice Work 5

* Планований час на реалізацію 2 год
* Важливі деталі для врахування в імплементації
* Врахувати використання цілочисельних значень в списку, 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.

Програма №9 Self Practice Work (Algotester lab3v1)

* Планований час на реалізацію 30 хв
* Важливі деталі для врахування в імплементації
* Врахувати, що якщо # іде під ряд, то це буде вважатись одним ритуалом.

Програма №10 Self Practice Work (Algotester lab6v3)

* Планований час на реалізацію 60 хв
* Важливі деталі для врахування в імплементації
* Перевести кожен елемент стрічки, котра задається одразу у число, як тільки вона вводиться.

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

Деталі по конфігурації середовища

Достатньо базових розширень та налаштувань Visual Studio Code, C++, нічого додаткового не потрібно для кожного завдання.

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

Завдання №1 VNS Lab 10

Деталі по програмі

Сформувати двонаправлений список. Знищити з нього елемент після елемента із заданим номером, додати К елементів у початок списку, виконати друк списку, записати його у файл, видалити список й виконати друк, відновити список з файлу та виконати його друк.

#include <iostream>

#include <vector>

#include <fstream>

using namespace std;

class Node

{

public:

int data;

Node\* next;

Node\* previous;

Node (int value) : data(value), next(nullptr), previous(nullptr)

{

}

};

class List

{

private:

Node\* head;

Node\* tail;

public:

List (): head(nullptr), tail(nullptr)

{

}

void create\_list()

{

head = tail = nullptr;

};

void add\_element(int K, vector <int> data)

{

for(int i = 0; i < K; ++i)

{

Node\* new\_node = new Node(data[i]);

if(head == nullptr)

{

head = tail = new\_node;

}

else

{

new\_node -> next = head;

new\_node -> previous = nullptr;

head -> previous = new\_node;

head = new\_node;

}

}

};

void print()

{

Node\* current = head;

if(current == nullptr)

{

cout << "The list is empty" << endl;

}

while(current != nullptr)

{

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

current = current -> next;

}

cout << endl;

};

void erase\_element(int K, int index)

{

Node\* current = head;

for(int i = 0; i < index && current; ++i)

{

current = current -> next;

}

for(int i = 0; i < K && current; ++i)

{

Node\* next\_node = current -> next;

Node\* previous\_node = current -> previous;

if(previous\_node)

{

previous\_node -> next = next\_node;

}

else

{

head = next\_node;

}

if(next\_node)

{

next\_node -> previous = previous\_node;

}

else

{

tail = next\_node;

}

delete current;

current = next\_node;

}

};

void write\_to\_file()

{

ofstream outf("list.txt");

Node\* current = head;

while(current != nullptr)

{

outf << current -> data << " ";

current = current -> next;

}

outf.close();

};

void kill\_list()

{

Node\* current = head;

Node\* next;

while(current != nullptr)

{

next = current -> next;

delete current;

current = next;

}

head = nullptr;

};

void staying\_alive()

{

create\_list();

ifstream inf("list.txt");

int number;

vector <int> data;

while(inf >> number)

{

data.push\_back(number);

}

inf.close();

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

add\_element(data.size(), data);

};

};

int main()

{

List new\_list;

new\_list.create\_list();

int K;

int times;

cout << "How many times do you want to add your numbers? \n It should be at least 2 times)" << endl;

cin >> times;

for(int i = 0; i < times; ++i)

{

cout << "How many numbers you want to add?" << endl;

cin >> K;

cout << "Please add your numbers" << endl;

vector <int> data;

for(int i = 0; i < K; ++i)

{

int number;

cin >> number;

data.push\_back(number);

}

new\_list.add\_element(K, data);

cout << "New list after adding: ";

new\_list.print();

}

int index;

cout << "What is the index of the element, you want to delete?" << endl;

cin >> index;

cout << "Please enter how many elements, you want to delete" << endl;

cin >> K;

new\_list.erase\_element(K, index);

cout << "New list after deleting : ";

new\_list.print();

new\_list.write\_to\_file();

new\_list.kill\_list();

cout << "list is R.I.P ((" << endl;

new\_list.print();

new\_list.staying\_alive();

cout << "I always come back..." << endl;

new\_list.print();

new\_list.kill\_list();

return 0;

}

Блок з кодом до завдання №1

Завдання №2 Algotester Lab 5

Деталі по програмі

Вивести N рядків по M елементів в рядку через пробіл - висоти карти, тобто вийде, так звана мапа, де висоти будуть на 1 зменшуватись з кожним віддаленням від вершини.

#include <iostream>

#include <cmath>

using namespace std;

int main()

{

int N, M, x, y;

cin >> N >> M >> x >> y;

int map[N][M];

x -= 1;

y -= 1;

int pick = 0;

int step;

int minimal = 0;

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

{

for(int j = 0; j < M; ++j)

{

if(i == x && j == y)

{

map[i][j] = pick;

}

else

{

step = abs(i - x) + abs(j - y);

map[i][j] = pick - step;

}

if(minimal > map[i][j])

{

minimal = map [i][j];

}

}

}

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

{

for(int j = 0; j < M; ++j)

{

cout << map[i][j] - minimal << " ";

}

cout << endl;

}

}

Блок з кодом до завдання №2

Завдання №3 Algotester Lab 7-8

Деталі по програмі

Реалізувати структуру даних двозв’язний список як шаблон класу. Вивести відповіді на запити вставки, видалення, визначення розміру, отримання значення елементу за індексом, модифікувати значення елементу за індексом, вивести список на екран у зазначеному в умові форматі.

#include <iostream>

#include <string>

using namespace std;

template <typename T>

class Node

{

public:

T data;

Node\* next;

Node\* previous;

Node(T value) : data(value), next(nullptr), previous(nullptr)

{

}

};

template <typename T>

class List

{

private:

Node<T>\* head;

Node<T>\* tail;

int count;

public:

List() : head(nullptr), tail(nullptr), count(0)

{

}

T get(int index)

{

Node<T>\* current = head;

for(int i = 0; i < index && current; ++i)

{

current = current -> next;

}

if(current)

{

return current -> data;

}

return T();

}

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

{

Node<T>\* new\_node;

Node<T>\* current = head;

for(int i = 0; i < index && current; ++i)

{

current = current -> next;

}

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

{

new\_node = new Node<T> (elements[i]);

if(!head)

{

head = tail = new\_node;

}

else

{

if(current)

{

new\_node -> next = current;

new\_node -> previous = current -> previous;

current -> previous = new\_node;

if(new\_node -> previous)

{

new\_node -> previous -> next = new\_node;

}

else

{

head = new\_node;

}

}

else

{

tail -> next = new\_node;

new\_node -> previous = tail;

tail = new\_node;

}

}

if(current)

{

current = new\_node -> next;

}

count++;

}

};

void print()

{

Node<T>\* current = head;

while(current)

{

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

current = current -> next;

}

cout << endl;

};

void erase(int index, int n)

{

Node<T>\* current = head;

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

{

current = current -> next;

}

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

{

Node<T>\* next\_node = current -> next;

Node<T>\* previous\_node = current -> previous;

if(previous\_node)

{

previous\_node -> next = next\_node;

}

else

{

head = next\_node;

}

if(next\_node)

{

next\_node -> previous = previous\_node;

}

else

{

tail = next\_node;

}

delete current;

current = next\_node;

}

count -= n;

}

void set(int index, T element)

{

Node<T>\* current = head;

for(int i = 0; i < index && current; ++i)

{

current = current -> next;

}

if(current)

{

current -> data = element;

}

};

int size()

{

return count;

};

};

int main()

{

int Q;

cin >> Q;

List<int> new\_list;

for(int commands = 0; commands < Q; commands++)

{

string option;

cin >> option;

if(option == "insert")

{

int index, N;

cin >> index >> N;

int\* elements = new int[N];

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

{

cin >> elements[i];

}

new\_list.insert(index, N, elements);

delete[] elements;

}

if(option == "print")

{

new\_list.print();

}

if(option == "get")

{

int index;

cin >> index;

cout << new\_list.get(index) << endl;

}

if(option == "erase")

{

int index, n;

cin >> index >> n;

new\_list.erase(index, n);

}

if(option == "set")

{

int index, element;

cin >> index >> element;

new\_list.set(index, element);

}

if(option == "size")

{

cout << new\_list.size() << endl;

}

}

return 0;

}

Блок з кодом до завдання №3

Завдання №4 Class Practice Work 1

Деталі по програмі

Програма реалізовує метод реверсу зв’язного списку.

#include <iostream>

#include <vector>

using namespace std;

class Node

{

public:

int data;

Node\* next;

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

{

}

};

class List

{

private:

Node\* head;

public:

List() : head(nullptr)

{

}

void create\_list(int amount, vector <int> data)

{

if (amount == 0 || data.empty())

{

return;

}

head = new Node(data[0]);

Node\* current = head;

for(int i = 1; i < amount; ++i)

{

Node\* new\_node = new Node(data[i]);

current -> next = new\_node;

current = new\_node;

}

};

void reverse()

{

Node\* previous = nullptr;

Node\* current = head;

Node\* next\_node = nullptr;

while(current != nullptr)

{

next\_node = current -> next;

current -> next = previous;

previous = current;

current = next\_node;

}

head = previous;

};

void printList()

{

Node\* current = head;

while(current != nullptr)

{

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

current = current -> next;

}

cout << endl;

};

};

int main()

{

int amount;

vector <int> data;

cout << "Enter the amount of your numbers: " << endl;

cin >> amount;

cout << "Enter numbers: " << endl;

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

{

int number;

cin >> number;

data.push\_back(number);

}

List new\_list;

new\_list.create\_list(amount, data);

cout << "Start list: ";

new\_list.printList();

new\_list.reverse();

cout << "Reverse list: ";

new\_list.printList();

return 0;

}

Блок з кодом до завдання №4

Завдання №5 Class Practice Work 2

Деталі по програмі

Програма реалізує метод порівняння зв’язних списків.

return 0;

#include <iostream>

#include <vector>

using namespace std;

class Node

{

public:

int data;

Node\* next;

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

{

}

};

class List

{

private:

Node\* head;

public:

List() : head(nullptr)

{

}

void create\_list(int amount, vector <int> data)

{

if (amount == 0 || data.empty())

{

return;

}

head = new Node(data[0]);

Node\* current = head;

for(int i = 1; i < amount; ++i)

{

Node\* new\_node = new Node(data[i]);

current -> next = new\_node;

current = new\_node;

}

};

void printList()

{

Node\* current = head;

while(current != nullptr)

{

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

current = current -> next;

}

cout << endl;

};

Node\* Head()

{

return head;

}

};

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

{

Node\* current1 = h1;

Node\* current2 = h2;

while(current1 != nullptr && current2 != nullptr)

{

if(current1 -> data != current2 -> data)

{

return false;

}

current1 = current1 -> next;

current2 = current2 -> next;

}

if(current1 != nullptr || current2 != nullptr)

{

return false;

}

return true;

};

int main()

{

int amount1, amount2;

vector <int> data1;

vector <int> data2;

cout << "Enter the amount of your numbers for list 1: " << endl;

cin >> amount1;

cout << "Enter numbers for list 1: " << endl;

for(int i = 0; i < amount1; ++i)

{

int number;

cin >> number;

data1.push\_back(number);

}

cout << "Enter the amount of your numbers for list 2: " << endl;

cin >> amount2;

cout << "Enter numbers for list 2: " << endl;

for(int i = 0; i < amount2; ++i)

{

int number;

cin >> number;

data2.push\_back(number);

}

List list1;

List list2;

list1.create\_list(amount1, data1);

list2.create\_list(amount2, data2);

if(compare(list1.Head(), list2.Head()))

{

cout << "List 1 and list 2 are similar" << endl;

}

else

{

cout << "They are different" << endl;

}

return 0;

}

Блок з кодом до завдання №5

Завдання №6 Class Practice Work 3

Деталі по програмі

Програма реалізує метод додавання великих чисел за допомогою зв’язного списку.

#include <iostream>

#include <vector>

using namespace std;

class Node

{

public:

int data;

Node\* next;

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

{

}

};

class List

{

private:

Node\* head;

public:

List() : head(nullptr)

{

}

void create\_list(int amount, vector <int> data)

{

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

{

Node\* new\_node = new Node(data[i]);

if(head == nullptr)

{

head = new\_node;

}

else

{

new\_node -> next = head;

head = new\_node;

}

}

};

void printList()

{

Node\* current = head;

while(current != nullptr)

{

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

current = current -> next;

}

cout << endl;

};

Node\* Head()

{

return head;

};

void reverse()

{

Node\* previous = nullptr;

Node\* current = head;

Node\* next\_node = nullptr;

while(current != nullptr)

{

next\_node = current -> next;

current -> next = previous;

previous = current;

current = next\_node;

}

head = previous;

};

};

vector <int> add(Node\* n1, Node\* n2)

{

Node\* current1 = n1;

Node\* current2 = n2;

vector <int> result;

int carry = 0;

while(current1 != nullptr || current2 != nullptr || carry != 0)

{

int sum = carry;

if(current1 != nullptr)

{

sum += current1 -> data;

current1 = current1 -> next;

}

if(current2 != nullptr)

{

sum += current2 -> data;

current2 = current2 -> next;

}

int digit = sum % 10;

carry = sum / 10;

result.insert(result.begin(), digit);

};

return result;

};

int main()

{

int amount1, amount2;

vector <int> data1;

vector <int> data2;

cout << "Enter the amount of your numbers for list 1: " << endl;

cin >> amount1;

cout << "Enter numbers for list 1: " << endl;

for(int i = 0; i < amount1; ++i)

{

int number;

cin >> number;

data1.push\_back(number);

}

cout << "Enter the amount of your numbers for list 2: " << endl;

cin >> amount2;

cout << "Enter numbers for list 2: " << endl;

for(int i = 0; i < amount2; ++i)

{

int number;

cin >> number;

data2.push\_back(number);

}

List number1;

List number2;

List final;

number1.create\_list(amount1, data1);

number2.create\_list(amount2, data2);

cout << "List with number 1: ";

number1.printList();

cout << "List with number 2: ";

number2.printList();

final.create\_list(add(number1.Head(), number2.Head()).size(), add(number1.Head(), number2.Head()));

cout << "List with the result: ";

final.printList();

cout << "List with the result, showed normally: " << endl;

final.reverse();

final.printList();

return 0;

}

Блок з кодом до завдання №6

Завдання №7 Class Practice Work 4

Деталі по програмі

Програма реалізовує метод віддзеркалення дерева.

#include <iostream>

using namespace std;

struct Node

{

int data;

Node\* left;

Node\* right;

Node(int value): data(value), left(nullptr), right(nullptr)

{

}

};

class BinaryTree

{

private:

Node\* root;

Node\* insert(Node\* node, int value)

{

if(node == nullptr)

{

return new Node(value);

}

if(value < node -> data)

{

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

}

if(value > node -> data)

{

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

}

return node;

};

public:

BinaryTree(): root(nullptr)

{

}

void insert(int value)

{

root = insert(root, value);

}

Node\* get\_root()

{

return root;

}

};

Node\* mirror\_flip(Node\* root)

{

if(root == nullptr)

{

return nullptr;

}

Node\* current = root -> left;

root -> left = mirror\_flip(root-> right);

root -> right = mirror\_flip(current);

return root;

};

void print\_tree(Node\* root, int level = 0)

{

if (root == nullptr)

{

return;

}

const int indentation = 4;

print\_tree(root->right, level + 1);

for (int i = 0; i < level \* indentation; ++i)

{

cout << " ";

}

cout << root->data << endl;

print\_tree(root->left, level + 1);

};

int main()

{

BinaryTree binary\_tree;

binary\_tree.insert(5);

binary\_tree.insert(3);

binary\_tree.insert(7);

binary\_tree.insert(2);

binary\_tree.insert(4);

binary\_tree.insert(6);

binary\_tree.insert(8);

cout << "Original tree:" << endl;

print\_tree(binary\_tree.get\_root());

Node\* mirrored\_root = mirror\_flip(binary\_tree.get\_root());

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

print\_tree(mirrored\_root);

return 0;

}

Блок з кодом до завдання №7

Завдання №8 Class Practice Work 5

Деталі по програмі

Програма реалізовує метод запису кожному батьківському вузлу дерева суму його підвузлів.

#include <iostream>

using namespace std;

struct Node

{

int data;

Node\* left;

Node\* right;

Node(int value): data(value), left(nullptr), right(nullptr)

{

}

};

class BinaryTree

{

private:

Node\* root;

Node\* insert(Node\* node, int value)

{

if(node == nullptr)

{

return new Node(value);

}

if(value < node -> data)

{

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

}

if(value > node -> data)

{

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

}

return node;

};

public:

BinaryTree(): root(nullptr)

{

}

void insert(int value)

{

root = insert(root, value);

}

Node\* get\_root()

{

return root;

}

};

int calculate(Node\* root)

{

if (root == nullptr)

{

return 0;

}

int left\_sum = calculate(root->left);

int right\_sum = calculate(root->right);

int total\_sum = left\_sum + right\_sum + root->data;

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

{

int temp = root->data;

root->data = left\_sum + right\_sum;

return total\_sum - temp;

}

return total\_sum;

};

void tree\_sum(Node\* root)

{

calculate(root);

};

void print\_tree(Node\* root, int level = 0)

{

if (root == nullptr)

{

return;

}

const int indentation = 4;

print\_tree(root->right, level + 1);

for (int i = 0; i < level \* indentation; ++i)

{

cout << " ";

}

cout << root->data << endl;

print\_tree(root->left, level + 1);

};

int main()

{

BinaryTree binary\_tree;

binary\_tree.insert(5);

binary\_tree.insert(3);

binary\_tree.insert(7);

binary\_tree.insert(2);

binary\_tree.insert(4);

binary\_tree.insert(6);

binary\_tree.insert(8);

cout << "Original tree:" << endl;

print\_tree(binary\_tree.get\_root());

tree\_sum(binary\_tree.get\_root());

cout << "New tree:" << endl;

print\_tree(binary\_tree.get\_root());

return 0;

}

Блок з кодом до завдання №8

Завдання №9 Self Practice Work (Algotester lab3v1)

Деталі по програмі

Програма виводить єдине ціле число — мінімальна кількість ритуалів, які потрібно провести, щоб відновити закляття.

#include <iostream>

#include <string>

using namespace std;

int main()

{

int N;

int counter = 0;

cin >> N;

cin.ignore();

string lines[N];

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

{

string line;

getline(cin, line);

lines[i] += line;

}

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

{

for(int j = 0; j < lines[i].size(); ++j)

{

if(lines[i][j] == '#')

{

counter++;

while(lines[i][j] == '#' && j < lines[i].size())

{

j++;

}

}

}

}

cout << counter << endl;

return 0;

}

Блок з кодом до завдання №9

Завдання №10 Self Practice Work (Algotester lab6v3)

Деталі по програмі

* Програма виводить Q разів відповідь у наступному форматі:
* Натуральне число M - кількість цифр, які можна вписати в клітинку
* M цифер розділених пробілом - можливі цифри.

#include <iostream>

#include <vector>

#include <algorithm>

#include <string>

#include <vector>

using namespace std;

int main()

{

int N, Q;

cin >> N;

int playground[N][N];

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

{

string row;

cin >> row;

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

{

playground[i][j] = row[j] - '0';

}

}

cin >> Q;

int questions[Q][2];

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

{

for (int j = 0; j < 2; ++j)

{

cin >> questions[i][j];

}

}

vector <int> answer;

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

{

int x = questions[i][0] - 1;

int y = questions[i][1] - 1;

int number = 1;

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

{

answer.push\_back(number);

number++;

}

if(playground[x][y] == 0)

{

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

{

if(playground[j][y] != 0)

{

answer.erase(remove(answer.begin(), answer.end(), playground[j][y]), answer.end());

}

if(playground[x][j] != 0)

{

answer.erase(remove(answer.begin(), answer.end(), playground[x][j]), answer.end());

}

}

}

else

{

vector<int> newAnswer;

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

{

if (answer[j] == playground[x][y])

{

newAnswer.push\_back(answer[j]);

}

}

answer = newAnswer;

}

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

for(int k = 0; k < answer.size(); ++k)

{

cout << answer[k] << " ";

}

cout << endl;

answer.clear();

}

return 0;

}

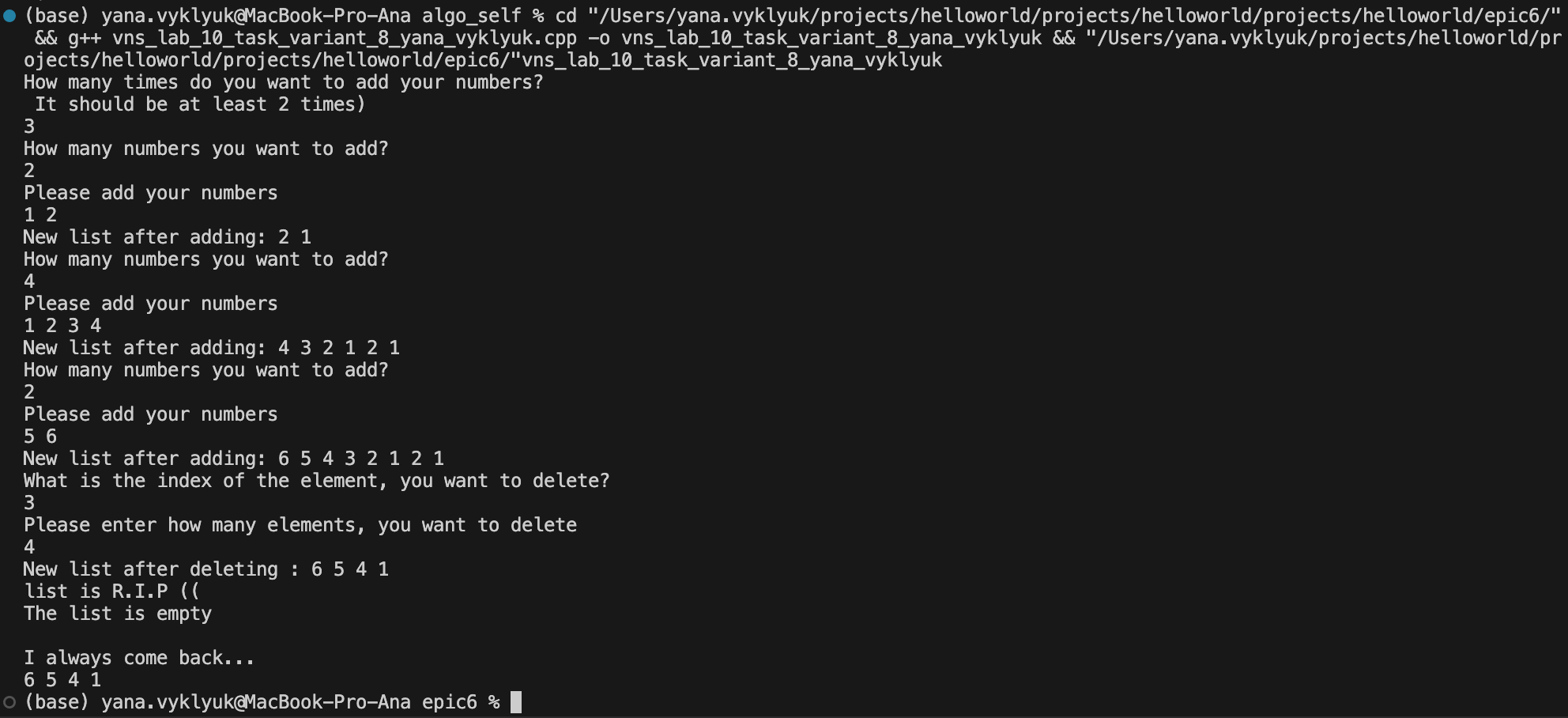
Блок з кодом до завдання №10

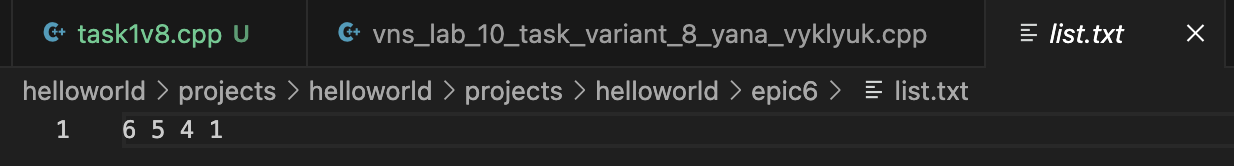
## **5. Результати виконання завдань, тестування та фактично затрачений час:**

Завдання №1 VNS Lab 10

Деталі по виконанню і тестуванню програми

Програма формує двонаправлений список з елементів, котрі вводить користувач. Знищує з нього ту кількість елементів, яку задає користувач, після елемента із номером, який теж задає користувач, додає К елементів у початок списку, виконує друк списку, запис його у файл, видаляє список й виконує його друк, відновлює список з файлу та виконує його друк.





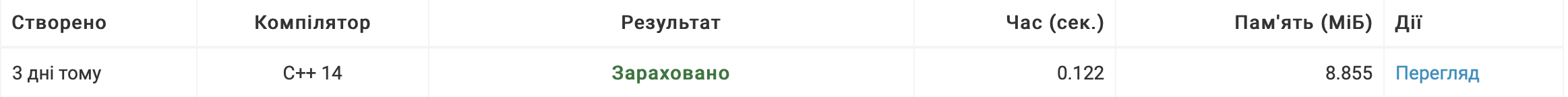
Блок з виконанням та тестуванням програми до завдання №1

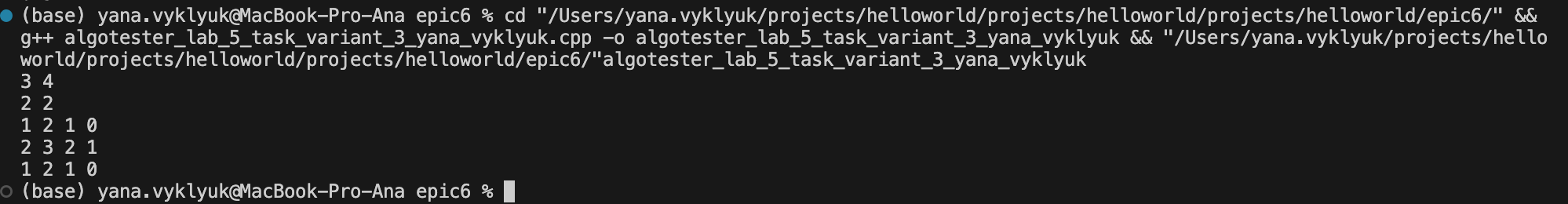
Час затрачений на виконання завдання 2 год

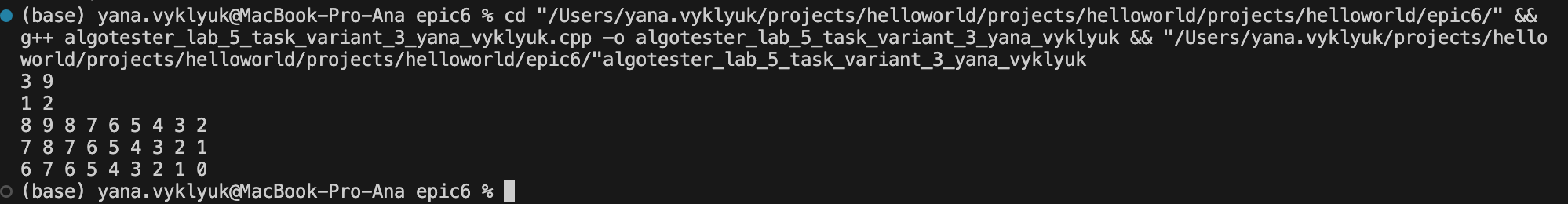
Завдання №2 Algotester Lab 5

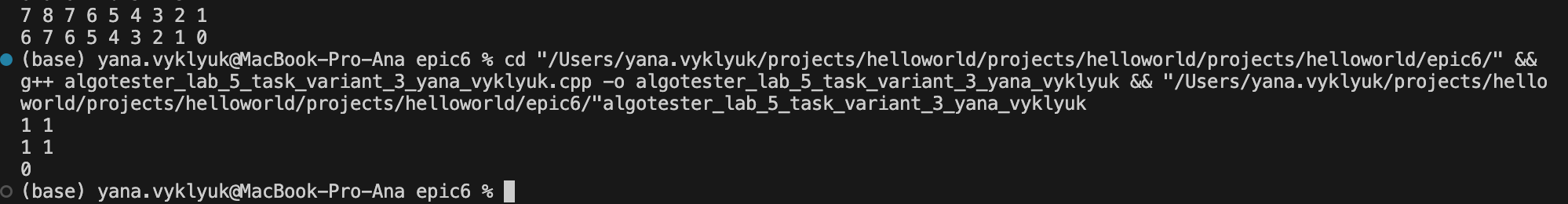
Деталі по виконанню і тестуванню програми

Код програми загружено на Algotester. Програма виводить N рядків по M елементів в рядку через пробіл - висоти карти, тобто так звану мапу, де висоти будуть на 1 зменшуватись з кожним віддаленням від вершини, доки мінімальна висота не буде дорівнювати 0.









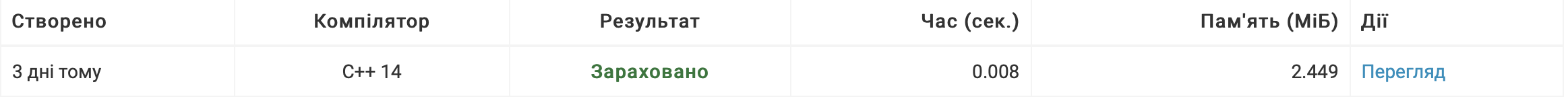
Блок з виконанням та тестуванням програми до завдання №2

Час затрачений на виконання завдання 90 хв

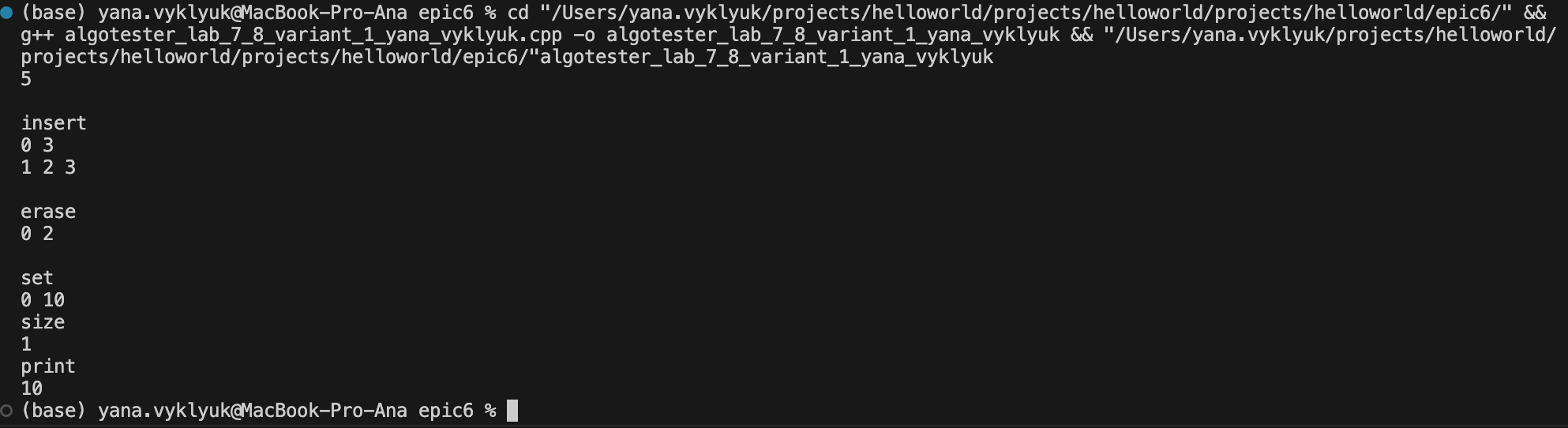
Завдання №3 Algotester Lab 7-8

Деталі по виконанню і тестуванню програми

Код програми загружено на Algotester. Програма реалізовує структуру даних двозв’язний список, виводить відповіді на запити вставки, видалення, визначення розміру, отримання значення елементу за індексом, модифікувати значення елементу за індексом, виводить список на екран у зазначеному в умові форматі. Всі запити реалізовані за допомогою методів класу. Програма написана як шаблон класу.







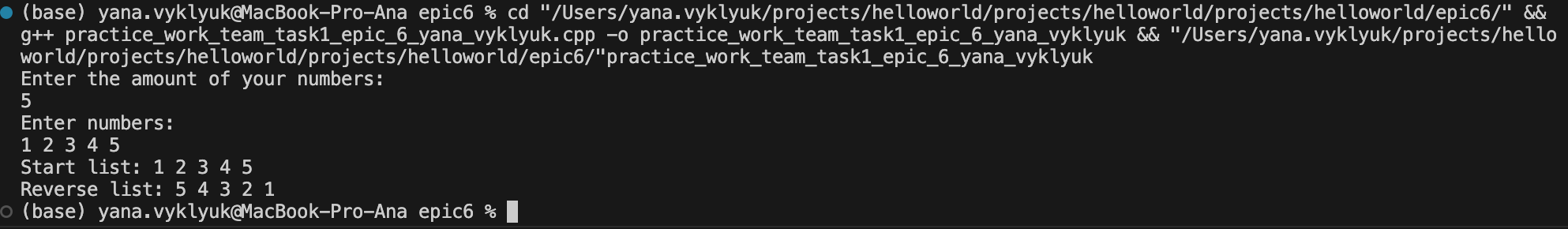
Блок з виконанням та тестуванням програми до завдання №3

Час затрачений на виконання завдання 2 год

Завдання №4 Class Practice Work 1

Деталі по виконанню і тестуванню програми

Програма реалізує метод реверсу зв’язного списку, використовує цілочисельні значення в списку, реалізовує допоміжний метод виведення вхідного і обернутого списків. Користувач вводить кількість елементів у списку, а потім самі елементи, програма виводить спочатку заданий список, потім реверсивний.



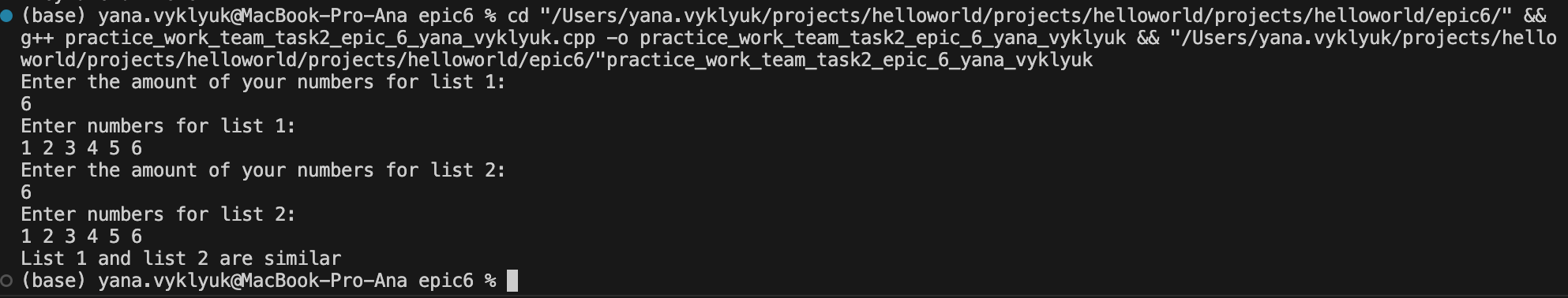
Блок з виконанням та тестуванням програми до завдання №4

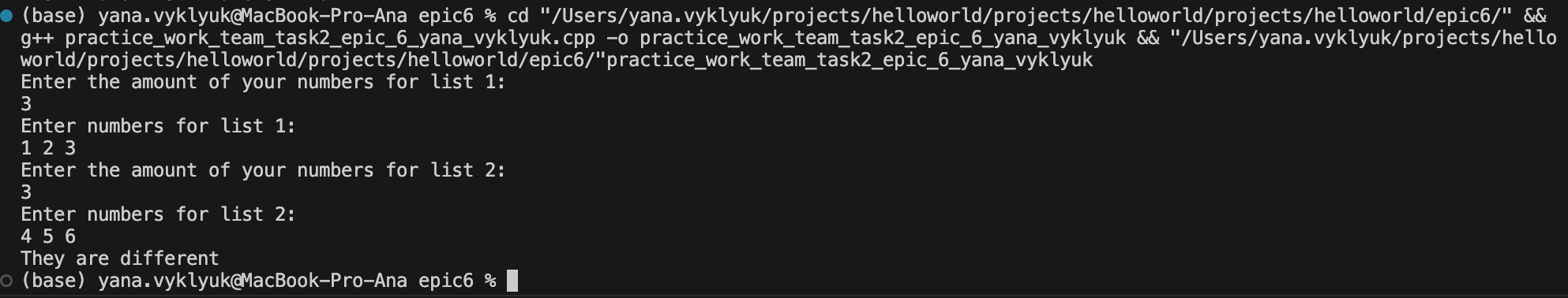
Час затрачений на виконання завдання 2 год.

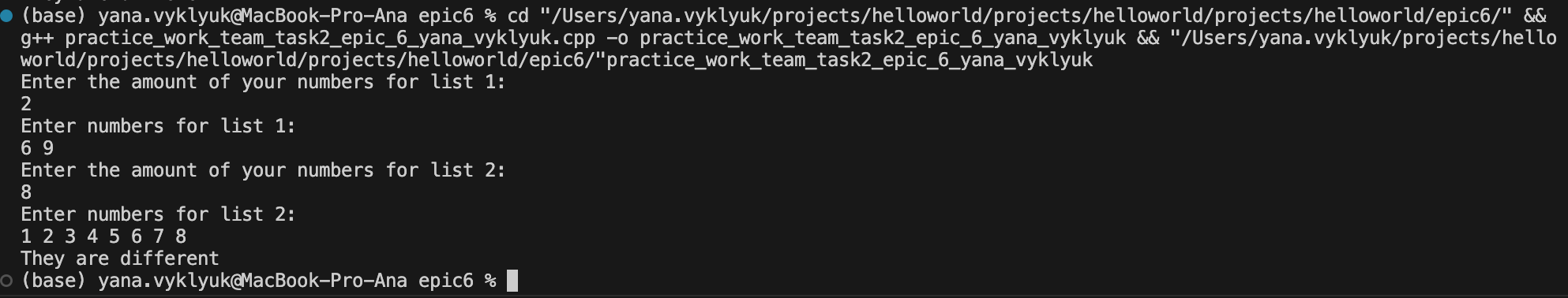
Завдання №5 Class Practice Work 2

Деталі по виконанню і тестуванню програми

Програма використовує цілочисельні значення в списку, реалізовує 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. Користувач вводить кількість елементів у першому списку та самі елементи, потім ідентично для другого списку, програма функцією перевіряє чи є ці списки однаковими(особливості зазначено вище) та виводить результат.







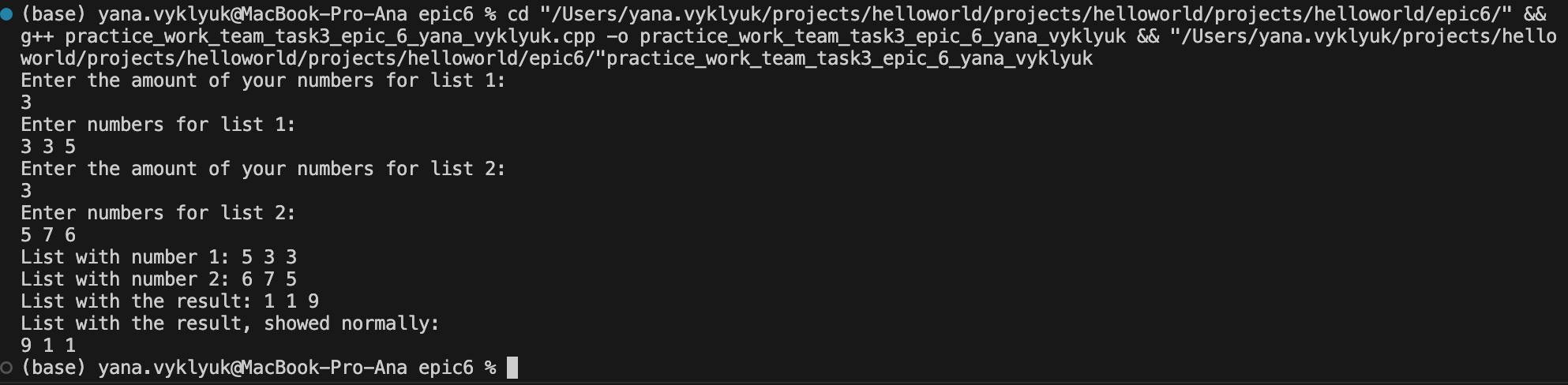
Блок з виконанням та тестуванням програми до завдання №2

Час затрачений на виконання завдання 90 хв.

Завдання №6 Class Practice Work 3

Деталі по виконанню і тестуванню програми

Програма використовує цифри від 0 до 9 для значень у списку; реалізовує 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. Користувач вводить кількість розрядів у першому та другому числі, потім самі числа, програма виводить списки із заданими цифрами чисел, котрі стоять реверсивно, потім виводить результат, який теж є реверсивним(від молодшого розряду до старшого), і виводить число у нормальному вигляді.



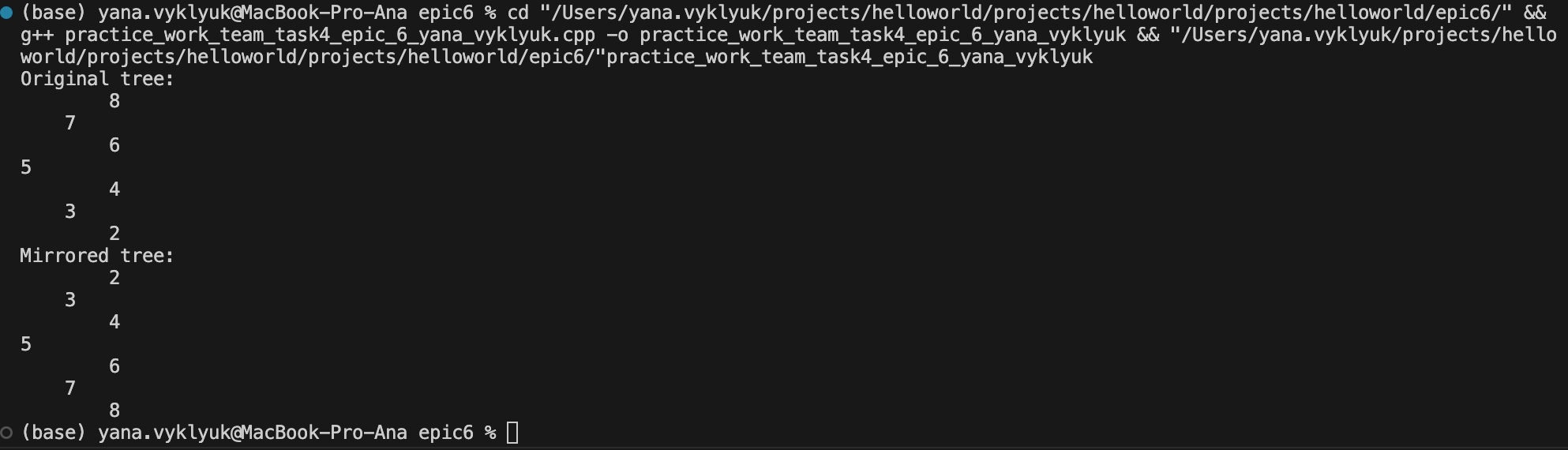
Блок з виконанням та тестуванням програми до завдання №6

Час затрачений на виконання завдання 2 год.

Завдання №7 Class Practice Work 4

Деталі по виконанню і тестуванню програми

Програма використовує цілочисельні значення в списку, реалізовує 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.



Блок з виконанням та тестуванням програми до завдання №7

Час затрачений на виконання завдання 2 год.

Завдання №8 Class Practice Work 5

Деталі по виконанню і тестуванню програми

Програма використовує цілочисельні значення в списку, реалізовує 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.



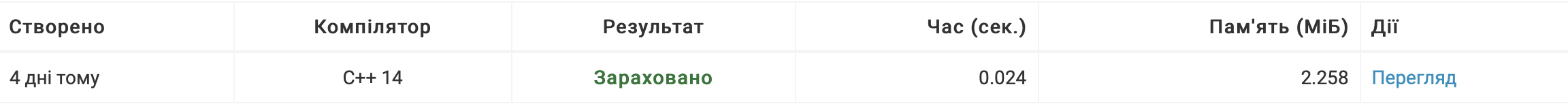
Блок з виконанням та тестуванням програми до завдання №8

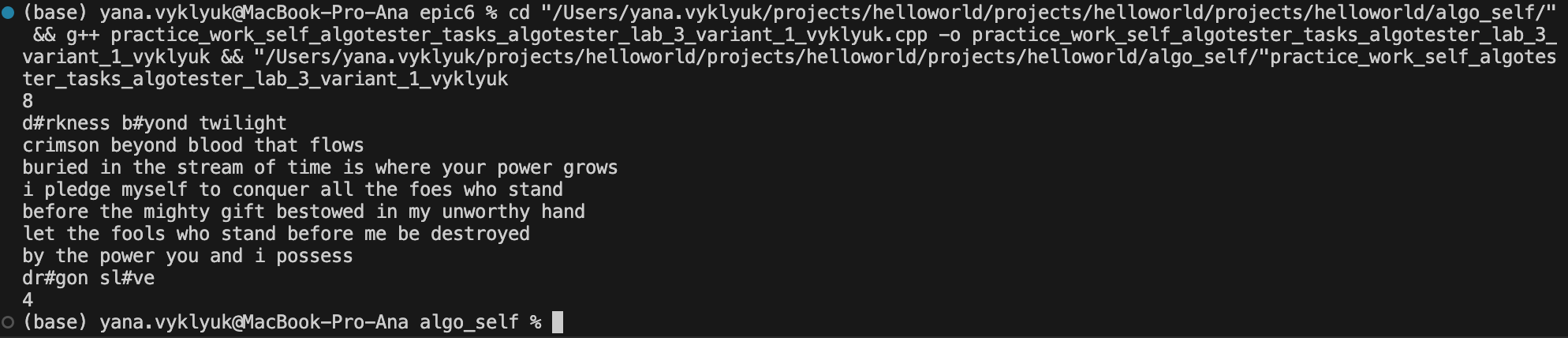
Час затрачений на виконання завдання 2 год.

Завдання №9 Self Practice Work (Algotester lab3v1)

Деталі по виконанню і тестуванню програми

Код програми загружено на Algotester. Програма враховує, що якщо # іде під ряд, то це буде вважатись одним ритуалом, виводить єдине ціле число — мінімальна кількість ритуалів, які потрібно провести, щоб відновити закляття.





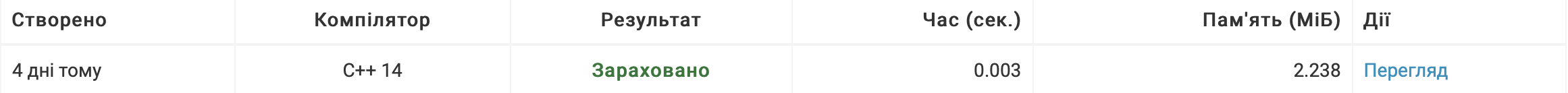
Блок з виконанням та тестуванням програми до завдання №9

Час затрачений на виконання завдання 30 хв.

Завдання №10 Self Practice Work (Algotester lab6v3)

Деталі по виконанню і тестуванню програми

* Програма переводить кожен елемент стрічки, котра задається одразу у число, як тільки вона вводиться, виводить Q разів відповідь у наступному форматі:
* Натуральне число M - кількість цифр, які можна вписати в клітинку
* M цифер розділених пробілом - можливі цифри.

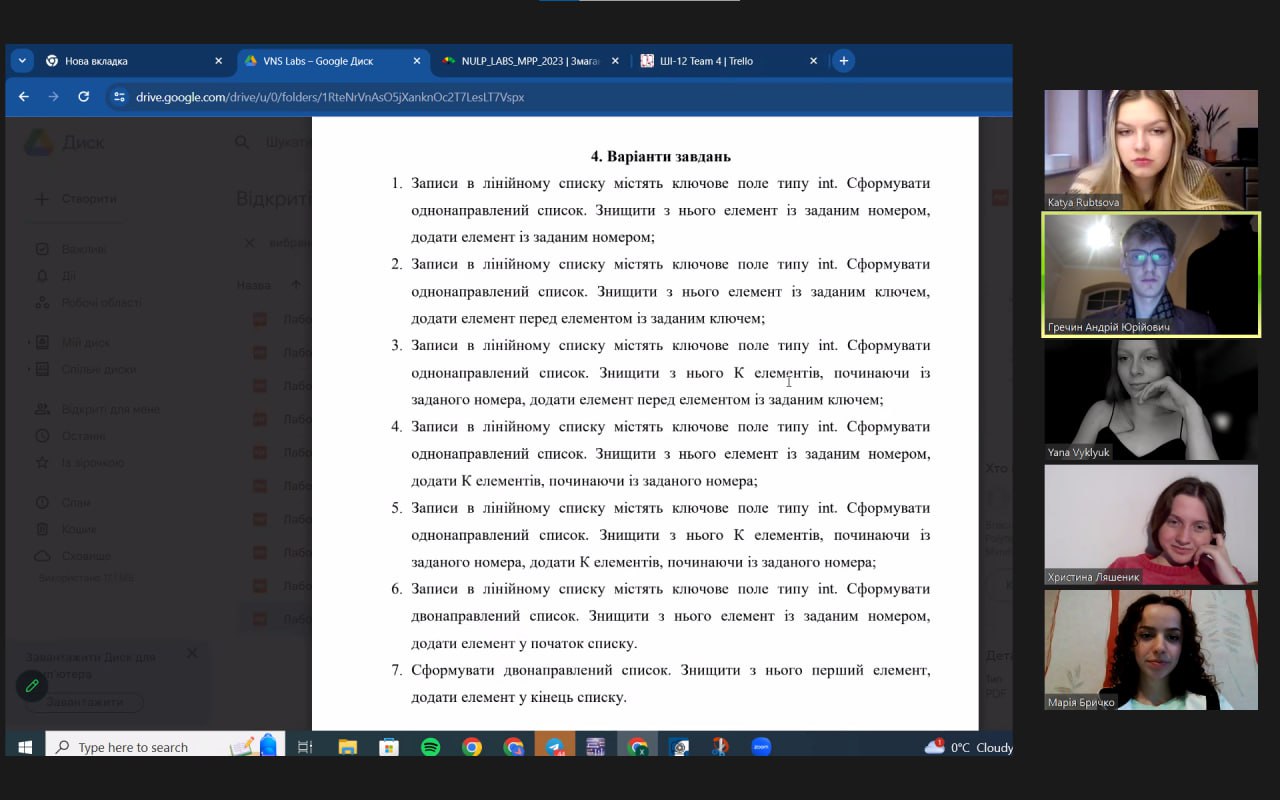


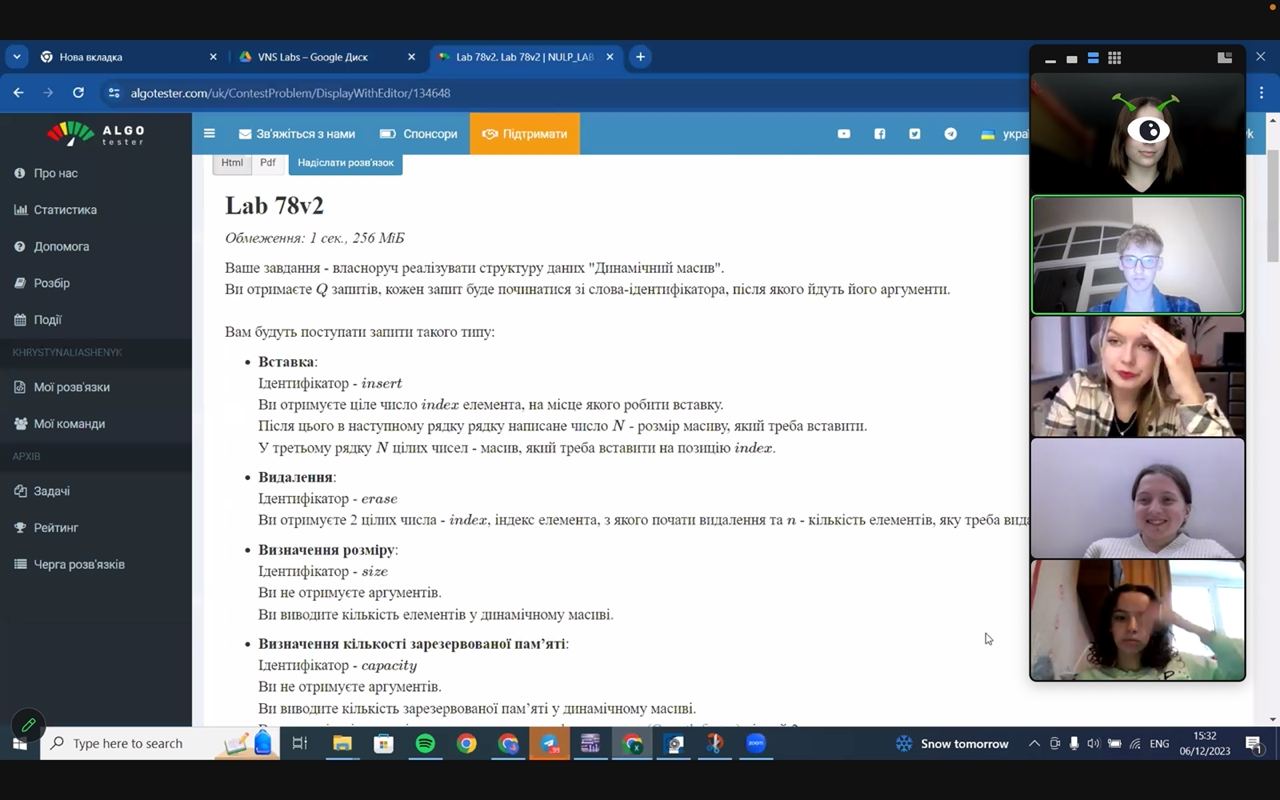


Блок з виконанням та тестуванням програми до завдання №10

Час затрачений на виконання завдання 60 хв.

## **6. Командна робота(мітинги, постановка задач та їх обговорення):**





# **Висновки:**

Ознайомилась та навчилась працювати з динамічними структурами даних, такі як Черга, Стек, Списки та Дерево, а також алгоритмами, що застосовуються для їх обробки. Отримала навички використання цих структур для ефективного управління даними в програмах.