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: «Основи програмування»

до:

ВНС Лабораторної Роботи № 10
Алготестер Лабораторної Роботи № 5
Алготестер Лабораторної Роботи № 7-8
Практичних Робіт до блоку № 6

Виконала:
Студентка групи ШІ-13
Кшик Олена Андріївна

Львів – 2024

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

Мета: Реалізувати різні динамічні структури і функції для роботи з ними.

Теоретичні відомості:

1. Основи Динамічних Структур Даних:
    - Вступ до динамічних структур даних: визначення та важливість
    - Виділення пам'яті для структур даних (stack і heap)
    - Приклади простих динамічних структур: динамічний масив
2. Стек:
    - Визначення та властивості стеку
    - Операції push, pop, top: реалізація та використання
    - Приклади використання стеку: обернений польський запис, перевірка балансу дужок
    - Переповнення стеку
3. Черга:
    - Визначення та властивості черги
    - Операції enqueue, dequeue, front: реалізація та застосування
    - Приклади використання черги: обробка подій, алгоритми планування
    - Розширення функціоналу черги: пріоритетні черги
4. Зв'язні Списки:
    - Визначення однозв'язного та двозв'язного списку
    - Принципи створення нових вузлів, вставка між існуючими, видалення, створення кільця(circular linked list)
    - Основні операції: обхід списку, пошук, доступ до елементів, об'єднання списків
    - Приклади використання списків: управління пам'яттю, FIFO та LIFO структури
5. Дерева:
    - Вступ до структури даних "дерево": визначення, типи
    - Бінарні дерева: вставка, пошук, видалення
    - Обхід дерева: в глибину (preorder, inorder, postorder), в ширину
    - Застосування дерев: дерева рішень, хеш-таблиці
    - Складніші приклади дерев: AVL, Червоно-чорне дерево
6. Алгоритми Обробки Динамічних Структур:
    - Основи алгоритмічних патернів: ітеративні, рекурсивні
    - Алгоритми пошуку, сортування даних, додавання та видалення елементів

Індивідуальний план опрацювання теорії:

- **Однозв'язний список**
- **Двозв'язний список**
- **Стек**
- **Черга**
- **Бінарне дерево**
- **Лекції і практичні заняття**

Виконання роботи:

1) Опрацювання завдання та вимог до програми та середовища

VNS Lab 10 Task 1 (23)

Написати програму, у якій створюються динамічні структури й виконати їхню обробку у відповідності зі своїм варіантом.

Для кожного варіанту розробити такі функції:

1. Створення списку.

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

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

4. Друк списку.

5. Запис списку у файл.

6. Знищення списку.

7. Відновлення списку з файлу.

23.Запису в лінійному списку містять ключове поле типу *char (рядок символів). Сформувати двонаправлений список. Знищити елемент із заданим ключем. Додати К елементів після елемента із заданим ключем.

Algotester Lab 5 Variant 2

В пустелі існує незвичайна печера, яка є двохвимірною. Її висота це N, ширина - M.
Всередині печери є пустота, пісок та каміння. Пустота позначається буквою O , пісок S і каміння X;
Одного дня стався землетрус і весь пісок посипався вниз. Він падає на найнижчу клітинку з пустотою, але він не може пролетіти через каміння.

Ваше завдання сказати як буде виглядати печера після землетрусу.

Algotester Lab 7-8 Variant 1

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 NN - розмір списку, який треба вставити.
  У третьому рядку NN 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** і-го елементу
  Ідентифікатор - get
  Ви отримуєте ціле число - index, індекс елемента.
  Ви виводите значення елемента за індексом.
- **Модифікація значення** і-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 <<

Class Practice Work

Задача №1 - Реверс списку (Reverse list)

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

Умови задачі:

- використовувати цілочисельні значення в списку;
- реалізувати метод реверсу;
- реалізувати допоміжний метод виведення вхідного і обернутого списків;

Задача №2 - Порівняння списків

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

Умови задачі:

- використовувати цілочисельні значення в списку;
- реалізувати функцію, яка ітеративно проходиться по обох списках і порівнює дані в кожному вузлі;
- якщо виявлено невідповідність даних або якщо довжина списків різна (один список закінчується раніше іншого), функція повертає ***false***.

Задача №3 – Додавання великих чисел

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

Умови задачі:

- використовувати цифри від 0 до 9 для значень у списку;
- реалізувати функцію, яка обчислює суму двох чисел, які збережено в списку; молодший розряд числа записано в голові списка (напр. $379 \implies 9 \to 7 \to 3$);
- функція повертає новий список, передані в функцію списки не модифікуються.

Задача №4 - Віддзеркалення дерева

TreeNode *create_mirror_flip(TreeNode *root);

Умови задачі:

- використовувати цілі числа для значень у вузлах дерева
- реалізувати функцію, що проходить по всіх вузлах дерева і міняє місцями праву і ліву вітки дерева
- функція повертає нове дерево, передане в функцію дерево не модифікується

Задача №5 - Записати кожному батьківському вузлу суму підвузлів

void tree_sum(TreeNode *root);

Умови задачі:

- використовувати цілочисельні значення у вузлах дерева;
- реалізувати функцію, яка ітеративно проходить по бінарному дереві і записує у батьківський вузол суму значень підвузлів
- вузол-листок не змінює значення
- значення змінюються від листків до кореня дерева

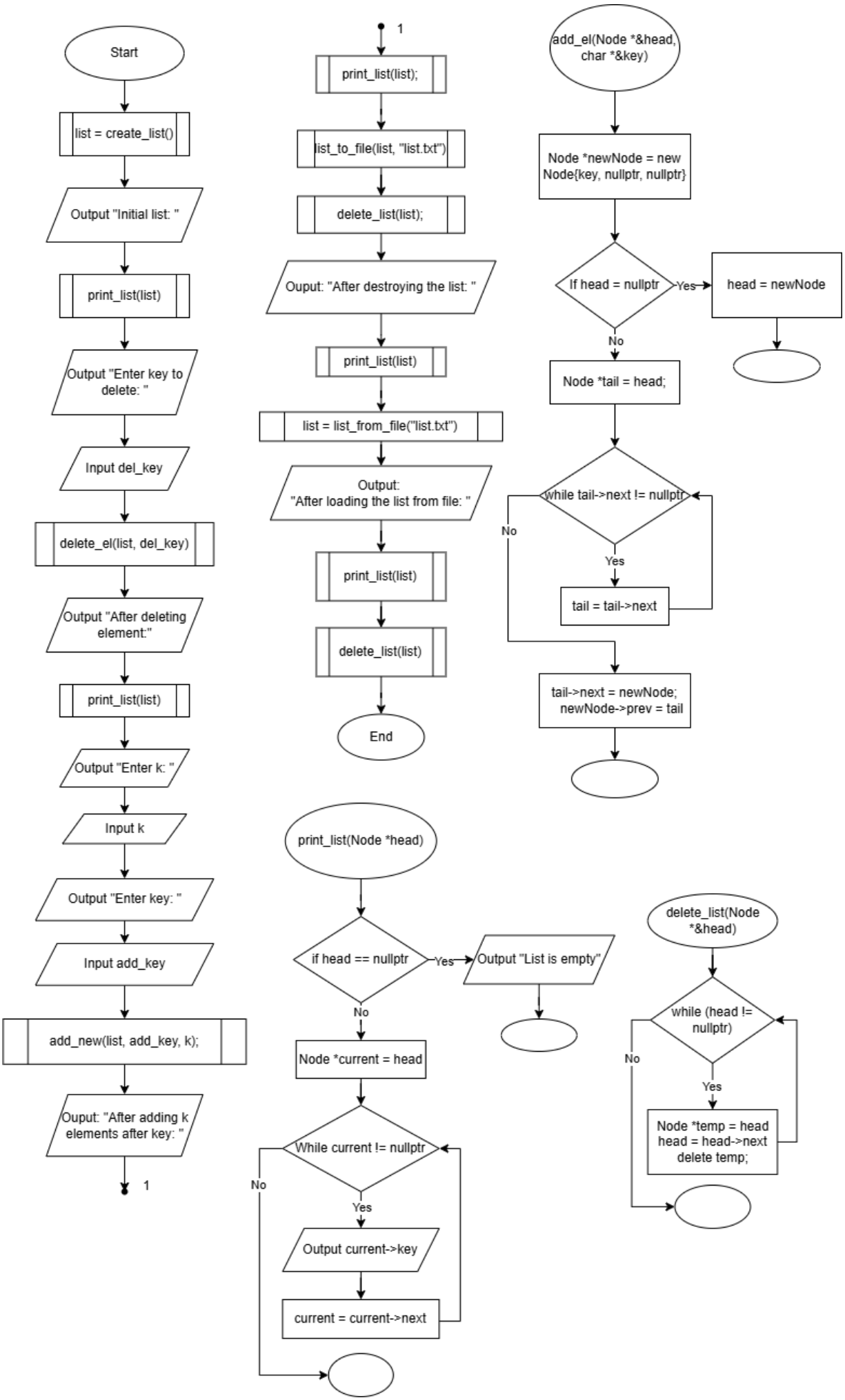
Self Practice Task:

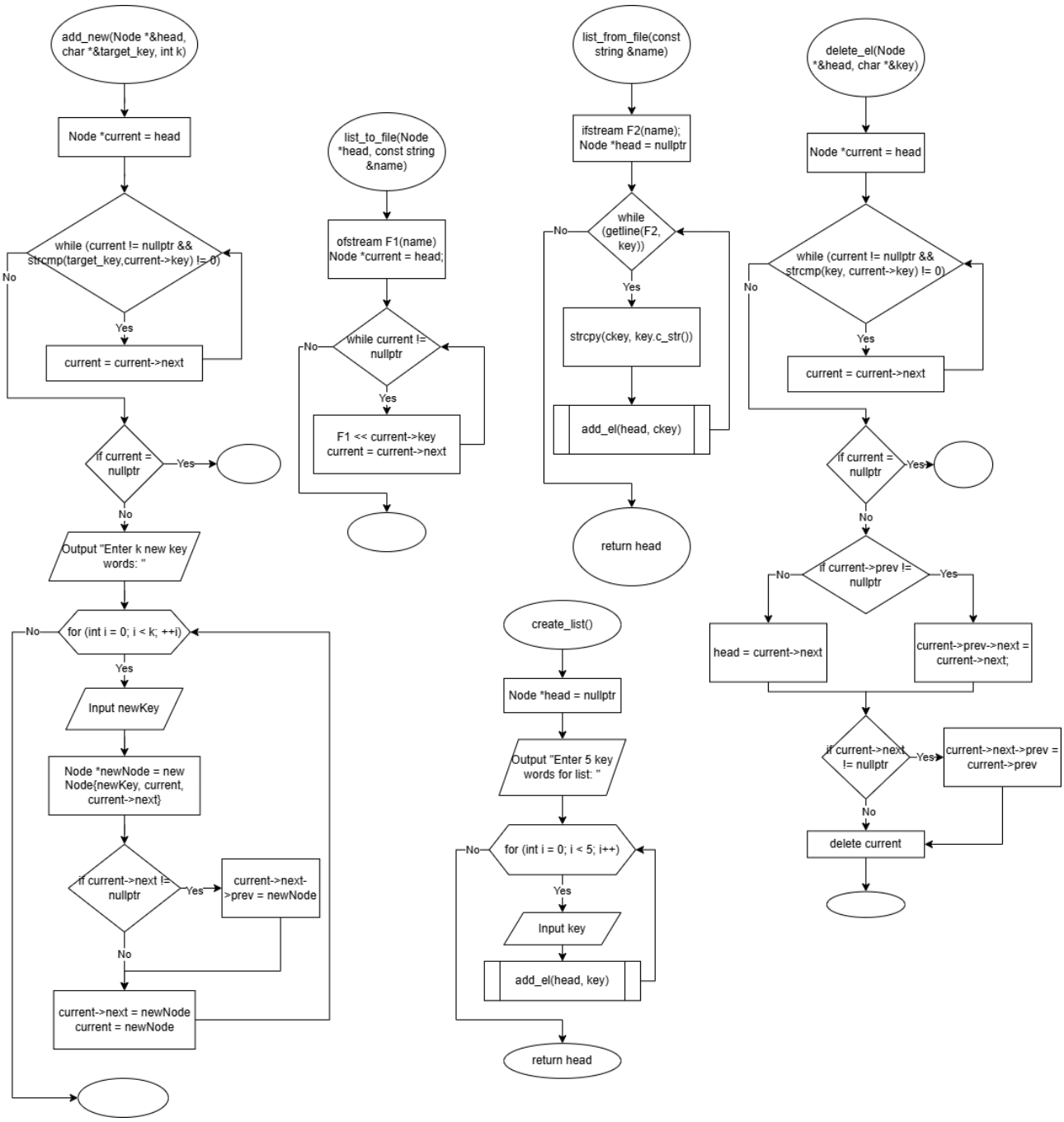
У вас є карта гори розміром N×M.
Також ви знаєте координати {x,y}, у яких знаходиться вершина гори.
Ваше завдання - розмалювати карту таким чином, щоб найнижча точка мала число 0, а пік гори мав найбільше число.

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.

2) Дизайн та планована оцінка часу виконання завдань

VNS Lab 10 Task 1 (23)


Start
list = create_list()
Output "Initial list: "
print_list(list)
Output "Enter key to delete: "
Input del_key
delete_el(list, del_key)
Output "After deleting element:"
print_list(list)
Output "Enter k: "
Input k
Output "Enter key: "
Input add_key
add_new(list, add_key, k);
Ouput: "After adding k elements after key: "
1
1
print_list(list);
list_to_file(list, "list.txt")
delete_list(list);
Ouput: "After destroying the list: "
print_list(list)
list = list_from_file("list.txt")
Output:
"After loading the list from file: "
print_list(list)
delete_list(list)
End
add_el(Node *&head, char *&key)
Node *newNode = new Node{key, nullptr, nullptr}
If head = nullptr
Yes
head = newNode
No
Node *tail = head;
while tail->next != nullptr
No
Yes
tail = tail->next
tail->next = newNode;
newNode->prev = tail
print_list(Node *head)
if head == nullptr
Yes
Output "List is empty"
No
Node *current = head
While current != nullptr
No
Yes
Output current->key
current = current->next
delete_list(Node *&head)
while (head != nullptr)
No
Yes
Node *temp = head
head = head->next
delete temp;



Плановий час виконання – 2 години.

Algotester Lab 5 Variant 2

Плановий час виконання – 30 хвилин.

Algotester Lab 78 Variant 1

Плановий час виконання – 1.5 години.

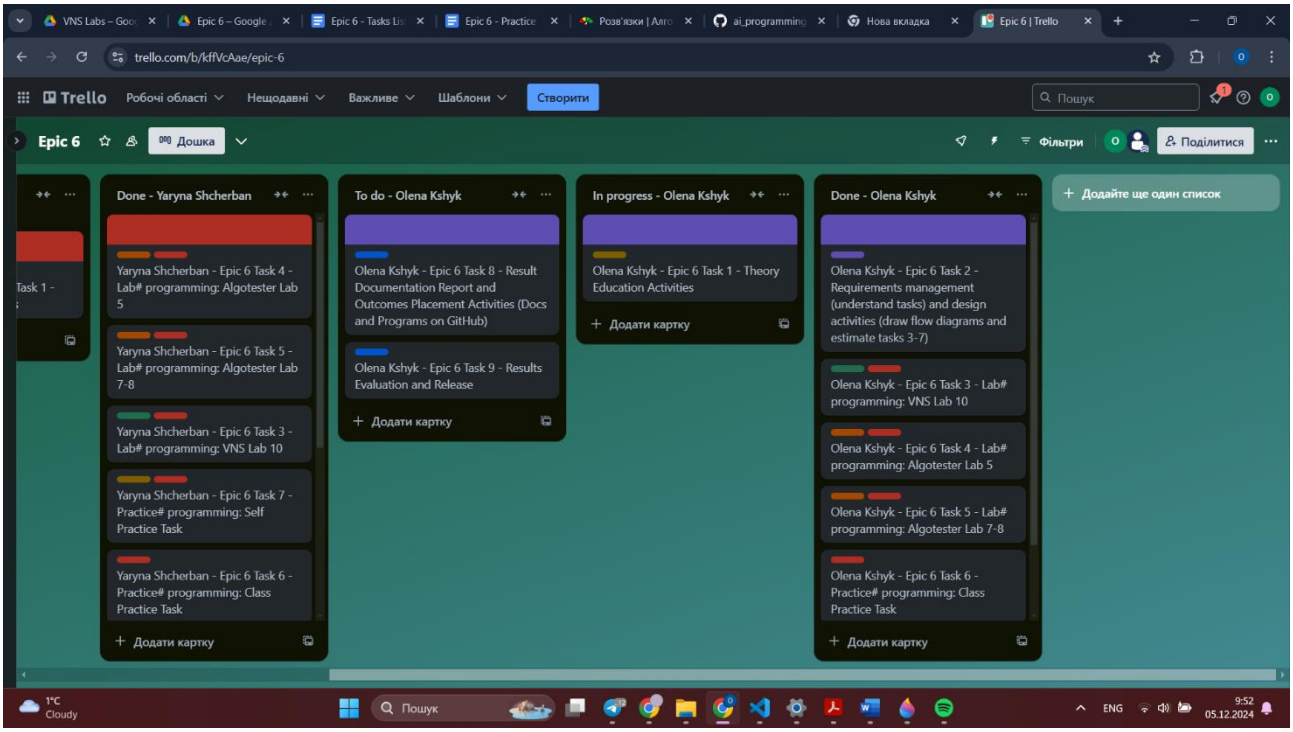
Class Practice Work

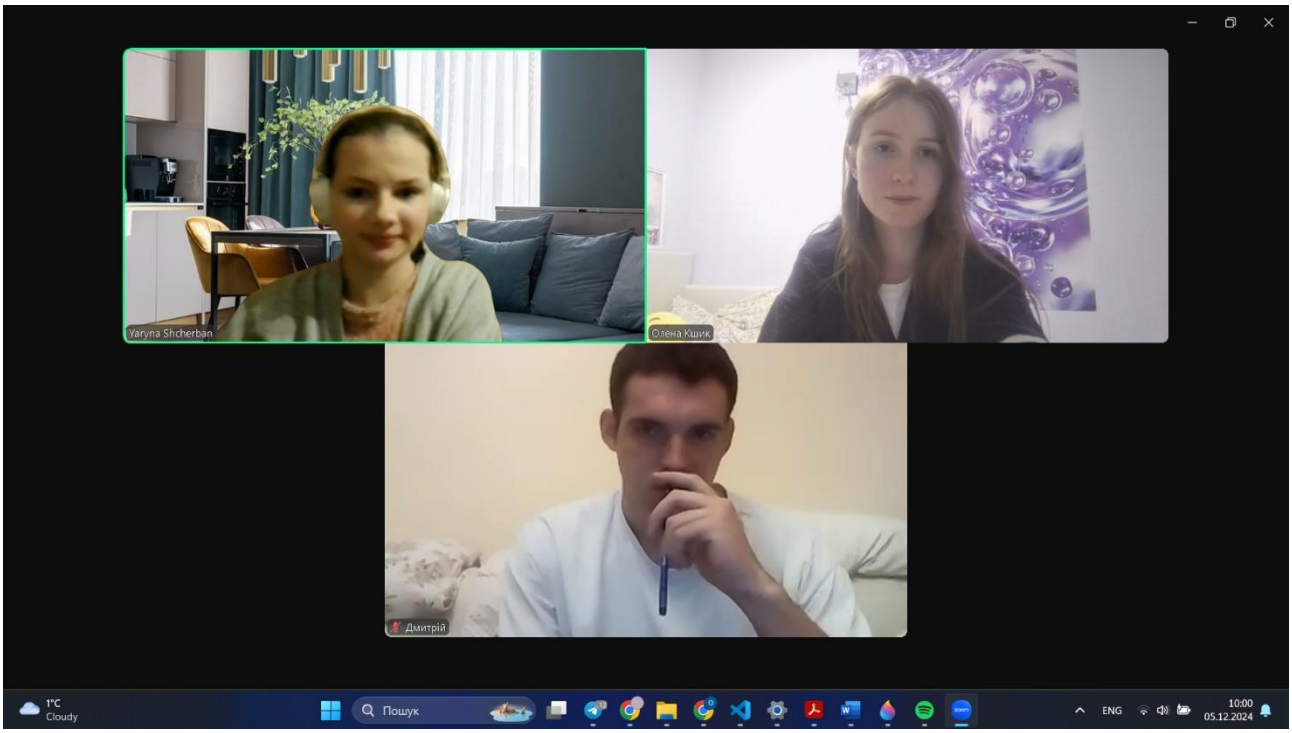
Плановий час виконання – 1.5 години.

Self Practice Task

Плановий час виконання – 30 хвилин.

3) Конфігурація середовища для виконання завдань





4) Код програми з посиланням на зовнішні ресурси

VNS Lab 10 Task 1: **vns_lab_10_task_1_variant_23_olena_kshyk.cpp**

Algotester Lab 5 Variant 2: **algotester_lab_5_task_2_olena_kshyk.cpp**

Algotester Lab 78 Variant 1: **algotester_lab_7_8_variant_1_olena_kshyk.cpp**

Class Practice Work 1-3: **practice_work_team_task_1_olena_kshyk.cpp**

Class Practice Work 4-5: **practice_work_team_task_2_olena_kshyk.cpp**

Self Practice Task: **practice_work_self_algotester_tasks_olena_kshyk.cpp**

5) Результати виконаних завдань, тестування та фактично затрачений час

VNS Lab 10 Task 1

```
Enter 5 key words for list:
first
second
third
fourth
fifth
Initial list:
first second third fourth fifth
Enter key to delete:
fourth
After deleting element:
first second third fifth
Enter k: 2
Enter key:
second
Enter 2 new key words:
element1
element2
After adding k elements after key:
first second element1 element2 third fifth
After destroying the list:
List is empty
After loading the list from file:
first second element1 element2 third fifth
```

Фактичний час виконання – 3 години.

Algotester Lab 5 Variant 2

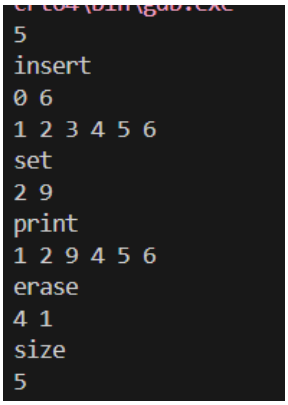
```
5 5
SSOSS
OOOOO
SOOXX
OOOOS
OOSOO

OOOOO
OOOSS
OOOXX
SOOOO
SSSOS
```

Фактичний час виконання – 30 хвилин.

4 хвилини тому	Lab 5v2 - Lab 5v2	C++ 23	Зараховано	0.025

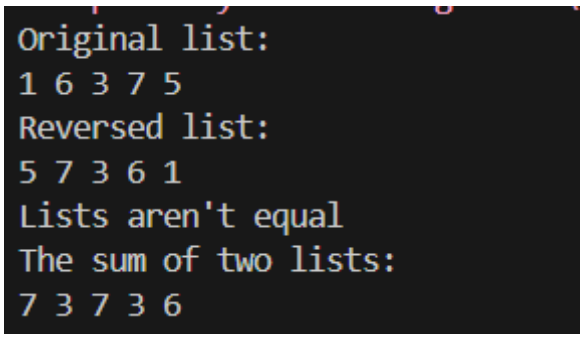
Algotester Lab 7-8 Variant 1

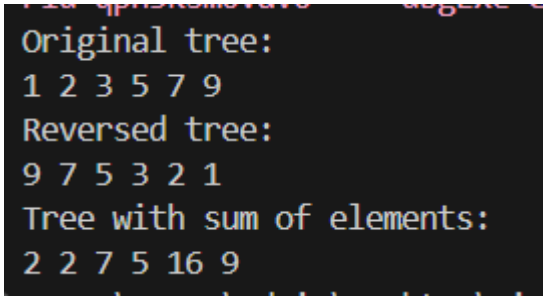


Фактичний час виконання – 1.5 години.

6 хвилин тому	Lab 78v1 - Lab 78v1	C++ 23	Зараховано	0.008

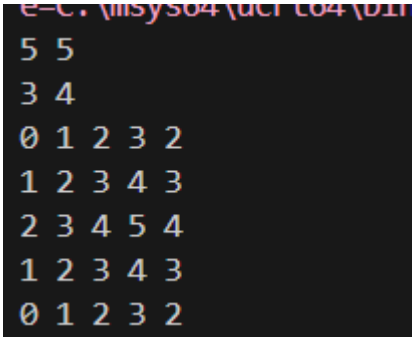
Class Practice Task





Фактичний час виконання – 2 години.

Self Practice Task



Фактичний час виконання – 40 хвилин.

4 хвилини тому	Lab 5v3 - Lab 5v3	C++ 23	Зараховано	0.106

Висновки: У ході лабораторної роботи я реалізувала деякі основні динамічні структури даних: списки та дерево, а також алгоритми для їх обробки. Отримані практичні навички дозволили закріпити розуміння принципів роботи цих структур і способів їх ефективного використання в різних задачах.