Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту

A blue and white logo

Description automatically generated

**Звіт**

**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

**Виконав:**

Студент групи ШІ-11

Цяпа Остап Андрійович

Львів 2024

**Тема роботи:**

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

**Мета роботи:**

Застосувати на практиці вивчений матеріал, реалізувати Linked List (Однозв’язний список), бінарне дерево.

**Теоретичні відомості:**

- Тема №1: Основи Динамічних Структур Даних.

- Тема №2: Стек.

- Тема №3: Черга.

- Тема №4: Зв’язні списки.

- Тема №5: Дерева.

- Тема №6: Алгоритми Обробки Динамічних Структур.

1. Індивідуальний план опрацювання теорії:

- Тема №1: Основи Динамічних Структур Даних:

○ Джерела інформації:

▪ Статті.

<https://www.youtube.com/watch?v=NyOjKd5Qruk&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=58>

- Що опрацьовано:

○ Вступ до динамічних структур.

○ Виділення пам’яті для структур даних (stack і heap)

○ Приклади простих динамічних структур

Запланований час на вивчення 40 хвилин

Витрачений час 40 хвилин.

  - Тема №2: Стек:

○ Джерела інформації:

▪ Статті.

<https://acode.com.ua/urok-111-stek-i-kupa/>

<https://www.youtube.com/watch?v=ZYvYISxaNL0&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=141>

- Що опрацьовано:

○ Визначення та властивості стеку

○ Операції push, pop, top: реалізація та використання

○ Переповнення стеку

Запланований час на вивчення 2 години.

Витрачений час 2 години.

- Тема №3: Черга:

○ Джерела інформації:

▪ Статті.

<https://www.youtube.com/watch?v=Yhw8NbjrSFA&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=142>

- Що опрацьовано

○ Визначення та властивості черги

○ Операціїї dequeue, front: реалізація та застосування

○ Приклади використання черги: обробка подій, алгоритми планування

○ Розширення функціоналу черги: пріоритети черги

Запланований час на вивчення 2 години.

Витрачений час 2 години.

- Тема №4: Зв’язні списки:

○ Джерела інформації:

▪ Статті.

https://www.youtube.com/watch?v=-25REjF\_atI&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=139

<https://www.youtube.com/watch?v=QLzu2-_QFoE&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=140>

- Що опрацьовано

○ Визначення однозв’язного та двозв’язного списку

○ Принципи створення нових вузлів, втсавка між існуючими, видалення, створення кільця (circular linked list)

○ Основні операції: обхід списку, пошук, доступ до елементів та об’єднання списків

○ Приклади використання списків: управління пам’яттю, FIFO та LIFO структури.

Запланований час на вивчення 2 години.

Витрачений час 2 години.

- Тема № 5: Дерева:

○ Джерела інформації:

▪ Статті.

https://www.youtube.com/watch?v=qBFzNW0ALxQ&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=144

- Що опрацьовано

○ 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

Запланований час на вивчення 2 години.

Витрачений час 2 години.

- Тема №6: Алгоритми Обробки Динамічних Структур:

○ Джерела інформації:

▪ Статті.

https://www.youtube.com/watch?v=mnwDpO4zqLA&t=433s

- Що опрацьовано

○ Основи алгоритмічних патернів: ітеративні, рекурсивні

○ Алгоритми пошуку, сортування даних, додавання та видалення елементів

Запланований час на вивчення 2 години.

Витрачений час 2 години.

Також користувався Microsoft Copilot який давав відповіді на конкретні питання по коду та теорії.

**Виконання роботи:**

**1. Опрацювання завдання до програм.**Завдання №1  
**VNS LAB 10 – TASK 1 (VARIANT 9)**

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.

Записи в лінійному списку містять ключове поле типу int. Сформувати

двонаправлений список. Знищити з нього К елементів перед елементом із

заданим номером, додати К елементів у кінець списку.

Завдання №2

**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 О , 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.

# **Вхідні дані**

У першому рядку 2 цілих числа N та M - висота та ширина печери

У N наступних рядках стрічка rowi яка складається з N цифер - i-й рядок матриці, яка відображає стан печери до землетрусу.

# **Вихідні дані**

N рядків, які складаються з стрічки розміром M - стан печери після землетрусу.

Завдання №3  
**ALGOTESTER LAB 7-8 (VARIANT 3)**

Ваше завдання - власноруч реалiзувати структуру даних "Двiйкове дерево пошуку".

Ви отримаєте Q запитiв, кожен запит буде починатися зi слова-iдентифiкатора, пiсля якого

йдуть його параметри.

Вам будуть поступати запити такого типу:

• Вставка:

Iдентифiкатор - insert

Ви отримуєте цiле число value - число, яке треба вставити в дерево.

• Пошук:

Iдентифiкатор - contains

Ви отримуєте цiле число value - число, наявнiсть якого у деревi необхiдно перевiрити.

Якщо value наявне в деревi - ви виводите Y es, у iншому випадку No.

• Визначення розмiру:

Iдентифiкатор - size

Ви не отримуєте аргументiв.

Ви виводите кiлькiсть елементiв у деревi.

• Вивiд дерева на екран

Iдентифiкатор - print

Ви не отримуєте аргументiв.

Ви виводите усi елементи дерева через пробiл.

Реалiзувати використовуючи перегрузку оператора <<

Вхiднi данi

Цiле число Q - кiлькiсть запитiв.

У наступних рядках Q запитiв у зазначеному в умовi форматi.

Вихiднi данi

Вiдповiдi на запити у зазначеному в умовi форматi.

Завдання №4  
**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;

### **Мета задачі**

***Розуміння структур даних:*** Реалізація методу реверсу для зв’язаних списків є чудовим способом для поглиблення розуміння зв’язаних списків як фундаментальної структури даних. Він заохочує практичний підхід до вивчення того, як структуруються пов’язані списки та як ними маніпулювати.  
**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***.

### **Мета задачі**

***Розуміння рівності в структурах даних:*** це завдання допомагає зрозуміти, як визначається рівність у складних структурах даних, таких як зв’язані списки. На відміну від примітивних типів даних, рівність пов’язаного списку передбачає порівняння кожного елемента та їх порядку.

**Задача №3 – Додавання великих чисел**

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

*Умови задачі:*

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

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

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

### Мета задачі

***Розуміння операцій зі структурами даних:*** це завдання унаочнює практичне використання списка для обчислювальних потреб. Арифметичні операції з великими числами це окремий клас задач, для якого використання списків допомагає обійти обмеження у представленні цілого числа сучасними комп’ютерами.  
**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

### **Мета задачі**

***Розуміння структур даних:*** Реалізація методу віддзеркалення бінарного дерева покращує розуміння структури бінарного дерева, виділення пам’яті для вузлів та зв’язування їх у єдине ціле. Це один з багатьох методів роботи з бінарними деревами.  
**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

### **Мета задачі**

***Розуміння структур даних:*** Реалізація методу підрахунку сум підвузлів бінарного дерева покращує розуміння структури бінарного дерева. Це один з багатьох методів роботи з бінарними деревами.

Завдання №5  
**SELF PRACTICE WORK ALGOTESTER**

Зеник та Марічка люблять шукати халяву. Тож ось вона.

У Зеника є n синіх кульок, на i-ій кульці записане число ai. Зенику цікаво, скількома способами він може пофарбувати деякі кульки в жовтий колір так, щоб усі числа, записані на жовтих кульках, були строго меншими за числа на синіх кульках.

Зауважте, що Зеник повинен пофарбувати хоча б одну кульку в жовтий колір, також він може пофарбувати всі кульки.

# **Вхідні дані**

У першому рядку задано одне ціле число n — кількість кульок.

У другому рядку задано n цілих чисел ai — числа, записані на кульках.

# **Вихідні дані**

У єдиному рядку виведіть одне ціле число — скількома способами він може пофарбувати деякі кульки в жовтий колір.

Завдання №6  
**SELF PRACTICE WORK ALGOTESTER**

Зеник та Марічка грають у поле чудес. Спочатку Зеник пише на дошці загадане слово й закриває всі його букви. За один хід Марічка називає букву, а Зеник відкриває всі такі букви у слові.

Вам необхідно визначити, за яку мінімальну кількість ходів Марічка зможе відкрити всі букви у слові.

Наприклад, якщо Зеник загадав слово MAMA, то Марічка зможе його відкрити за два ходи, назвавши букви M та A.

# **Вхідні дані**

Вхідні дані містять рядок s — загадане Зеником слово.

# **Вихідні дані**

В одному рядку виведіть ціле число — мінімальну кількість ходів.

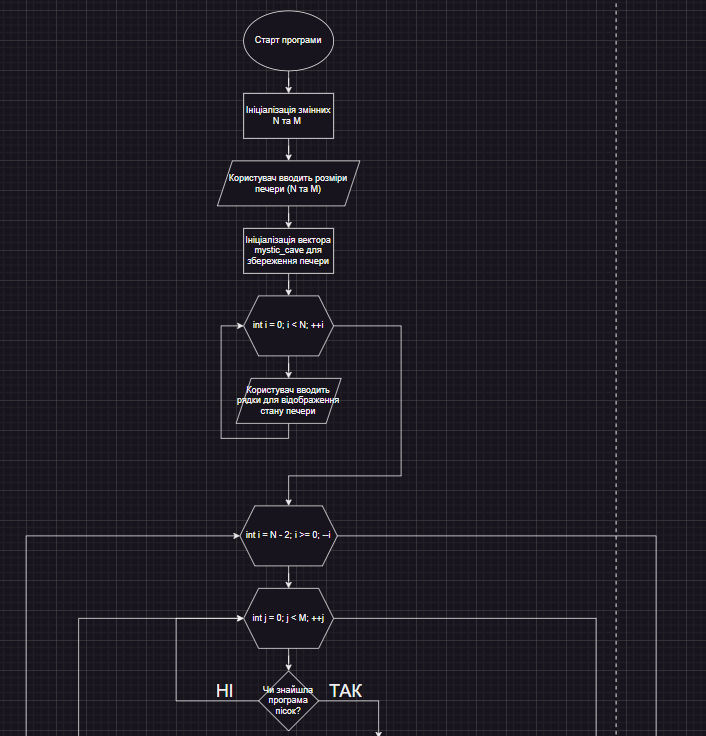
**2. Вимоги та планувальна оцінка часу виконання завдань:**

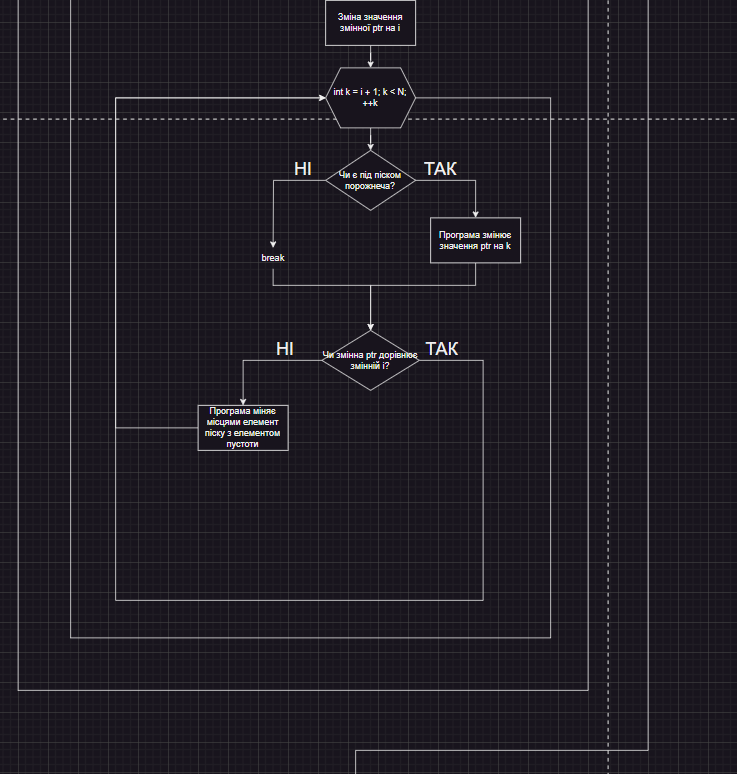
**Програма №1**

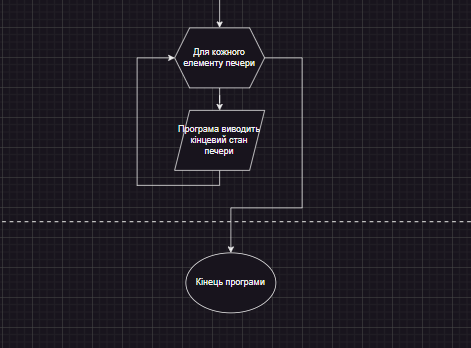
* Важливі деталі для реалізації програми.
* Усі операції з однозв'язним списком передбачають використання динамічної пам’яті (new і delete), тому важливо уникати витоків пам’яті, видаляючи вузли після видалення або очищення списку. Використовувати біблотеку fstream для запису даних у файл.
* Плановий час на реалізацію 3 години.

**Програма №2**

* Важливі деталі для реалізації програми.
* Блок – схема





  
*Рисунок 2.1. Блок – схема до програми 2*

* Використовувати для реалізації матрицю чарів(вектор у векторі) а також цикли, для того щоб проходитися по стовпцях та рядках матриці.
* Плановий час на реалізацію 2 години.

**Програма №3**

* Важливі деталі для реалізації програми.
* Зрозуміти, що таке Linked List (Однозв’язний список) і навчитися його реалізовувати за допомогою вказівників та структури і перетворити це все у шаблон класу.
* Плановий час на реалізацію 5 годин.

**Програма №4**

* Важливі деталі для реалізації програми.
* Зрозуміти, що таке бінарне дерево пошуку та навчитися його реалізовувати.
* Плановий час на реалізацію 5 годин.

**Програма №5**

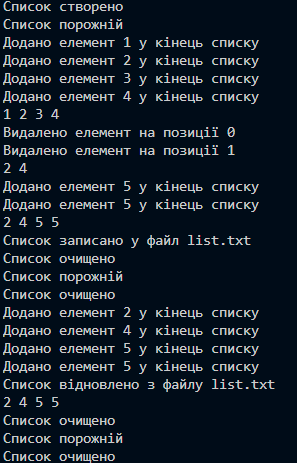
* Важливі деталі для реалізації програми.
* Використовувати для реалізації бібліотеку set для швидкого вирішення.
* Плановий час на реалізацію 2 години.

**3. Код програм з посиланням на зовнішні ресурси та фактично затрачений час:**

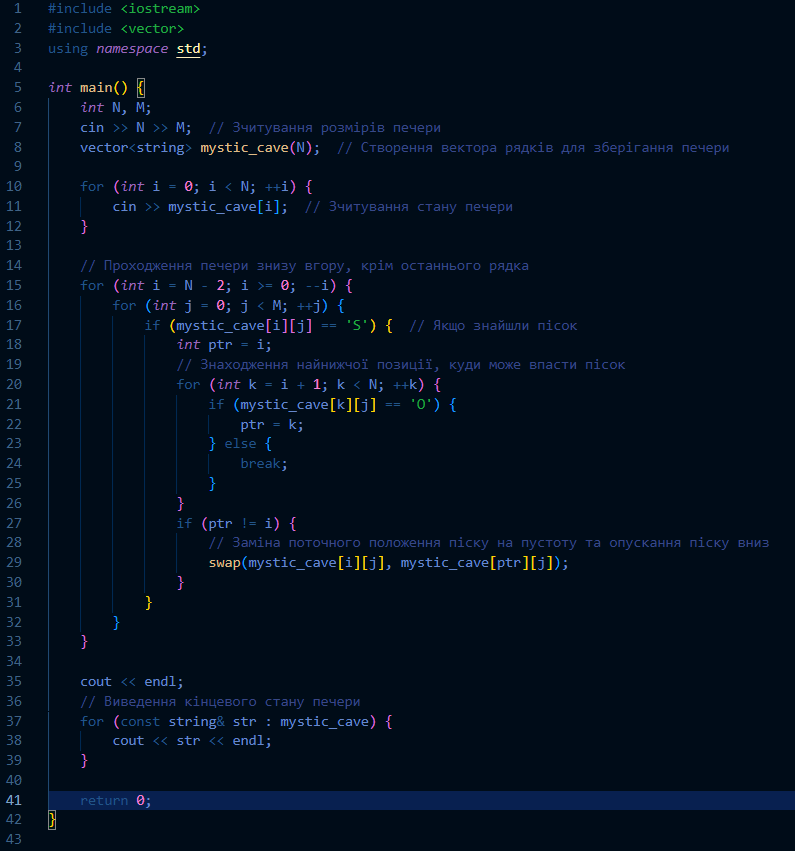
**Завдання №1  
**

****

*Рисунок 3.1. Код до програми № 1*

  
*Рисунок 3.2. Приклад виконання програми № 1*

Фактично затрачений час 4 години.

**Завдання №2  
***Рисунок 3.3. Код до програми № 2*

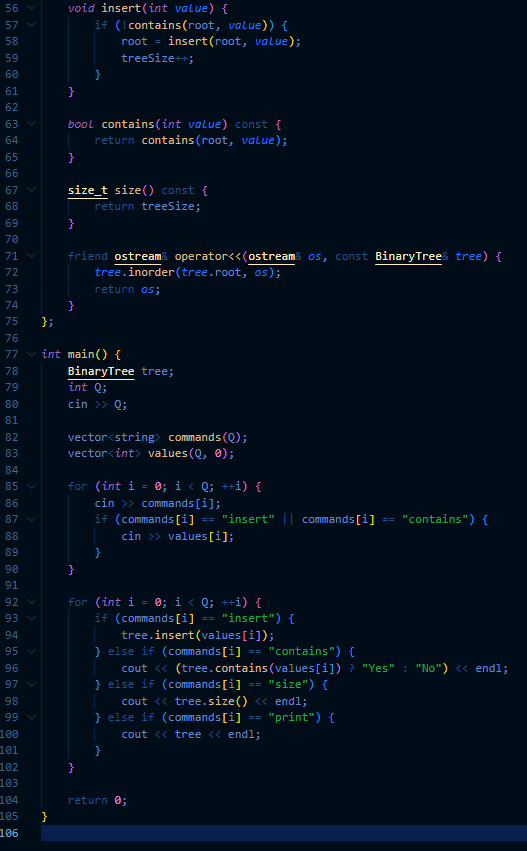
  
*Рисунок 3.4. Приклад виконання програми № 2*

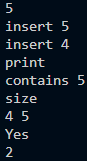
  
*Рисунок 3.5. Статус задачі на алготестері*

Фактично затрачений час 2 години.

**Завдання №3**



*Рисунок 3.6. Код до програми № 3*

*  
Рисунок 3.7. Приклад виконання програми №3*

*  
Рисунок 3.8. Статус задачі на Algotester*

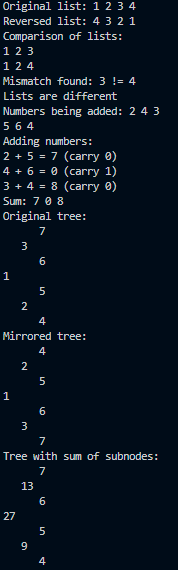
Фактично затрачений час 4 години.

**Завдання №4**

****

****

*****Рисунок 3.9. Код до завдання номер 4*

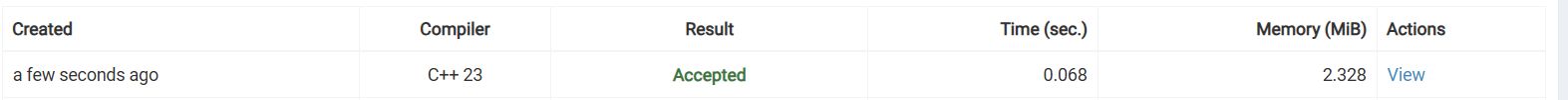
*****Рисунок 3.10. Приклад виконання завдання номер 4*

Фактично затрачений час 5 годин.

**Завдання №5**

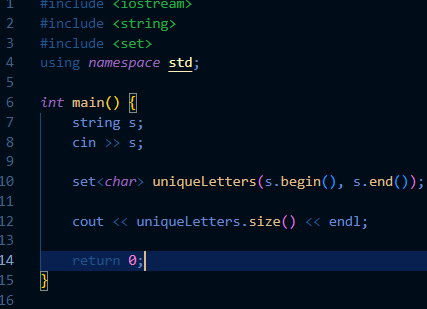
*****Рисунок 3.19. Код до програми №5*

*****Рисунок 3.20. Приклад виконання програми номер 5*

*****Рисунок 3.21. Статус задачі на Algotester*

Фактично затрачений час 20 хвилин.

**Завдання №6**

*****Рисунок 3.22. Код до програми №6*

*****Рисунок 3.23. Приклад виконання програми номер 5*

*****Рисунок 3.24. Статус задачі на Algotester*

Фактично затрачений час 15 хвилин.

**Посилання на пул реквест:** [Epic 6 - Ostap Tsiapa by Ostap2007ter · Pull Request #669 · artificial-intelligence-department/ai\_programming\_playground\_2024](https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/669)

**Висновок:** У рамках практичних і лабораторних робіт блоку №6 я засвоїв безліч нового матеріалу, включаючи динамічні структури даних, такі як черги, стеки, списки та дерева, а також алгоритми для їх обробки. На практиці я працював із бінарними деревами, створював зв'язані списки та застосовував алгоритм BFS для розв'язання задач.